REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(MM/DD/YYYY)* March 2014	**2. REPORT TYPE** Doctoral Thesis		**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE A Higher-Order Conservation Element Solution Element Method for Solving Hyperbolic Differential Equations on Unstructured Meshes			**5a. CONTRACT NUMBER**
			5b. GRANT NUMBER
			5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) David Bilyeu			**5d. PROJECT NUMBER**
			5e. TASK NUMBER
			5f. WORK UNIT NUMBER Q0AE
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFRL/RQRS 1 Ara Drive Edwards AFB, CA 93524			**8. PERFORMING ORGANIZATION REPORT NO.**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RQR 5 Pollux Drive Edwards AFB, CA 93524			**10. SPONSOR/MONITOR'S ACRONYM(S)**
			11. SPONSOR/MONITOR'S REPORT NUMBER(S) **AFRL-RQ-ED-OT-2014-030**
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited			
13. SUPPLEMENTARY NOTES Dissertation for presentation to University of Ohio Dissertation Committee PA Case Number: 14066; Clearance Date: 2/7/2014			

14. ABSTRACT

This dissertation presents an extension of the Conservation Element Solution Element (CESE) method from second- to higher-order accuracy. The new method retains the favorable characteristics of the original second-order CESE scheme, including (i) the use of the space-time integral equation for conservation laws, (ii) a compact mesh stencil, (iii) the scheme will remain stable up to a CFL number of unity, (iv) a fully explicit, time-marching integration scheme, (v) true multidimensionality without using directional splitting, and (vi) the ability to handle two- and three-dimensional geometries by using unstructured meshes. This algorithm has been thoroughly tested in one, two and three spatial dimensions and has been shown to obtain the desired order of accuracy for solving both linear and non-linear hyperbolic spatial differential equations. The scheme has also shown its ability to accurately resolve discontinuities in the solutions.

Higher order unstructured methods such as the Discontinuous Galerkin method and the Special Volume methods have been developed for one, two- and three-dimensional application. Although these schemes have seen extensive development and use, certain drawbacks of these methods have been well documented. For example, the explicit versions of these two methods have very stringent stability criteria. This stability criteria requires that the time step be reduced as the order of the solver increases, for a given simulation on a given mesh.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Jean-Luc Cambier
a. REPORT **Unclassified**	**b. ABSTRACT** **Unclassified**	**c. THIS PAGE** **Unclassified**	SAR	232	**19b. TELEPHONE NO** *(include area code)* 661 275-5649

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

A HIGHER-ORDER CONSERVATION ELEMENT SOLUTION ELEMENT METHOD FOR SOLVING HYPERBOLIC DIFFERENTIAL EQUATIONS ON UNSTRUCTURED MESHES

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of

Philosophy in the Graduate School of the Ohio State University

By

David Bilyeu, B.S., M.S.

Graduate Program in Mechanical Engineering

The Ohio State University

2013

Dissertation Committee:

Sheng-Tao John Yu, Ph.D., Advisor

Jean-Luc Cambier, Ph.D.

Chau-Lyan Chang, Ph.D.

Sin-Chung Chang, Ph.D.

A. Terrence Conlisk, Ph.D.

Datta Gaitonde, Ph.D.

Ahmet Selamet, Ph.D.

ABSTRACT

This dissertation presents an extension of the Conservation Element Solution Element (CESE) method from second- to higher-order accuracy. The new method retains the favorable characteristics of the original second-order CESE scheme, including (i) the use of the space-time integral equation for conservation laws, (ii) a compact mesh stencil, (iii) the scheme will remain stable up to a CFL number of unity, (iv) a fully explicit, time-marching integration scheme, (v) true multidimensionality without using directional splitting, and (vi) the ability to handle two- and three-dimensional geometries by using unstructured meshes. This algorithm has been thoroughly tested in one, two and three spatial dimensions and has been shown to obtain the desired order of accuracy for solving both linear and non-linear hyperbolic partial differential equations. The scheme has also shown its ability to accurately resolve discontinuities in the solutions.

Higher order unstructured methods such as the Discontinuous Galerkin[1] (DG) method and the Spectral Volume[2] (SV) methods have been developed for one-, two- and three-dimensional application. Although these schemes have seen extensive development and use, certain drawbacks of these methods have been well documented.

For example, the explicit versions of these two methods have very stringent stability criteria. This stability criteria requires that the time step be reduced as the order of the solver increases, for a given simulation on a given mesh.

The research presented in this dissertation builds upon the work of Chang[3], who developed a fourth-order CESE scheme to solve a scalar one-dimensional hyperbolic partial differential equation. The completed research has resulted in two key deliverables. The first is a detailed derivation of a high-order CESE methods on unstructured meshes for solving the conservation laws in two- and three-dimensional spaces. The second is the code implementation of these numerical methods in a computer code. For code development, a one-dimensional solver for the Euler equations was developed. This work is an extension of Chang's work on the fourth-order CESE method for solving a one-dimensional scalar convection equation. A generic formulation for the nth-order CESE method, where $n \geq 4$, was derived. Indeed, numerical implementation of the scheme confirmed that the order of convergence was consistent with the order of the scheme. For the two- and three-dimensional solvers, SOLVCON was used as the basic framework for code implementation. A new solver kernel for the fourth-order CESE method has been developed and integrated into the framework provided by SOLVCON. The main part of SOLVCON, which deals with unstructured meshes and parallel computing, remains intact. The SOLVCON code for data transmission between computer nodes for High Performance Computing (HPC).

To validate and verify the newly developed high-order CESE algorithms, several

one-, two- and three-dimensional simulations where conducted. For the arbitrary order, one-dimensional, CESE solver, three sets of governing equations were selected for simulation: (i) the linear convection equation, (ii) the linear acoustic equations, (iii) the nonlinear Euler equations. All three systems of equations were used to verify the order of convergence through mesh refinement. In addition the Euler equations were used to solve the Shu-Osher and Blastwave problems. These two simulations demonstrated that the new high-order CESE methods can accurately resolve discontinuities in the flow field. For the two-dimensional, fourth-order CESE solver, the Euler equation was employed in four different test cases. The first case was used to verify the order of convergence through mesh refinement. The next three cases demonstrated the ability of the new solver to accurately resolve discontinuities in the flows. This was demonstrated through: (i) the interaction between acoustic waves and an entropy pulse, (ii) supersonic flow over a circular blunt body, (iii) supersonic flow over a guttered wedge. To validate and verify the three-dimensional, fourth-order CESE solver, two different simulations where selected. The first used the linear convection equations to demonstrate fourth-order convergence. The second used the Euler equations to simulate supersonic flow over a spherical body to demonstrate the scheme's ability to accurately resolve shocks. All test cases used are well known benchmark problems and as such, there are multiple sources available to validate the numerical results. Furthermore, the simulations showed that the high-order CESE solver was stable at a CFL number near unity.

This work is dedicated to

friends and family, and the prospect of a vacation

ACKNOWLEDGMENTS

There are many people to whom I owe a great deal of thanks for their assistance during my time as a graduate student.

First I would like to thank my advisor, Professor John Yu, for his guidance and assistance during my time at The Ohio State University. In particular his assistance in ensuring that my work was well presented and understandable to a broader community was invaluable.

I would also like to thank my Air Force sponsor Dr. Jean-Luc Cambier for providing me the opportunity to work at the Air Force Lab and providing me the resources required to complete my PhD research.

I would also like to thank Dr. Chau-Lyan Chang, Dr Sin-Chung Chang, Dr. Terrence Conlisk, Dr. Datta Gaitonde, and Dr. Ahmet Selamet, for serving on my committee.

This work would have taken much longer if not for the help of my various labmates at Ohio State. In particular, I would like to thank Dr. Yung-Yu Chen for the development of SOLVCON, his knowledge of the CESE method and his ability to clarify many of the finer points of proper code development. I would also like to thank Dr. Lixiang Yang for his probing questions that made me realize that I did not know as much as I though I did.

I would also like to thank my coworkers at the In Space Propulsion branch at the Air Force Research Lab. They fostered a work environment that allowed me look past the two hour round trip commute and enjoy coming to work each day – well almost every day. In particular I would like to thank: Dr. Lord Cole for teaching me the skills required to get work done while working for the government, and for his attempt to get an out-of-shape grad student back into something that resembles a fit individual. Dr. Kooj, for without his planning of weekend events designed to get people out of the AV, I may have finished months earlier but would have found my life rather dull. Dr. Rob Martin for providing a liberal amount of cynicism to the lunch time conversations, and for his ability to find holes in my logic. Dr. Carl Ledderman for proving that not all mathematicians are difficult to understand. And to Mr. Hai Le, for the useful discussions in areas outside of my field of study.

There are also entities outside of school and work that have provided a great deal of help during my tenure as a graduate student. First is the websites wikipedia, and mathworld.wolfram, which I could almost always rely upon to provide the formulas for equations which I uncommonly use and commonly forget. The various stackexchange websites whose contributors have provided a wealth of solutions to a wide variety of programing problems. The creators of versions control which have saved me much grief over the last several years and will continue to do so for many years to come.

I need to thank the various DOD supercomputing facilities for granting the computing hours necessary to complete my numerous CFD simulations.

Finally and most importantly I would like to thanks my parents and my sister whose understanding and support allowed me to focus on completing my research.

VITA

2002–2006	B.S. in Aerospace Engineering, California State Polytechnic University, Pomona, California, USA
2006–2008	M.S. in Mechanical Engineering, The Ohio State University, Columbus, Ohio, USA

PUBLICATIONS

D. L. Bilyeu, S.-T. J. Yu, Y.-Y. Chen, and J.-L. Cambier, A two-dimensional fourth-order unstructured-meshed Euler solver based on the CESE method, Journal of Computational Physics, vol. 257, pp. 981–999, Jan. 2014.

FIELDS OF STUDY

Major Field: Mechanical Engineering

Specialization: Computational Mechanics and Fluid Dynamics

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This dissertation reports the development of a new high-order, unstructured-meshed, Computational Fluid Dynamics, CFD, method for time-accurate solutions of hyperbolic Partial Differential Equations, PDEs, as well as conservation laws. In addition to a detailed derivation of the algorithms, the associated numerical implementation of the algorithms have been developed and verified in one-, two-, and three-dimensions. These new high-order methods are an extension of the second-order, space-time Conservation-Element Solution-Element, CESE, method, originally developed by Chang[4, 5].

The objectives of this research are (i) to develop a suite of high-order algorithms based on the CESE method that are capable of solving fully coupled, linear and non-linear, hyperbolic PDEs on one-, two-, and three-dimensional unstructured meshes; (ii) for the newly developed numerical algorithms are generic and applicable to all hyperbolic systems, i.e. not tied to any single constitutive relation and/or physical process; (iii) to preserve all favorable attributes of the original second-order CESE method in the new high-order method, including: the stability constraint of the Courant-Friedrichs-Lewy, CFL, ≤ 1, the compact mesh stencil, a genuine multiple dimensional algorithm, and the use of the unique space-time integration equation instead of the Reynolds transport theorem; (iv) and to implement and verify that the algorithms behave as intended when used to simulate one-, two-, and three-dimensional flows, i.e. that the observed convergence rate matches the intended one and that the scheme accurately resolves flow discontinuities.

The rest of Chapter 1 is organized as follows. Section 1.1 reviews the different types of CFD solvers. Section 1.2 reviews three existing high-order CFD methods. Section 1.3 provides background information on the second-order CESE method. Section 1.4 provides more detailed discussions of the goals and scope of the present research. This chapter concludes with an outline of the remainder of this dissertation.

1.1 Review of High-Order CFD Solvers

The use of computers to simulate fluid flow has been in development for decades. During this period, CFD methodology has undergone steady improvements from the early first-order structured-meshed solvers to the latest high-order, unstructured-meshed solvers. This continuous development of CFD technologies has been driven by an increased reliance on computer simulations by engineers to aid in product development. CFD simulations have also been used by researchers to gain a better understanding of the physical processes observed in experiments when experimental measurement are not easily obtained.

CFD solvers are easily categorized by their order of accuracy and the type of mesh that they are designed to run on and. The order of the solver can be either low- or high-order; for a low-order solver the convergence rate is less than or equal to two. The two most common mesh types are structured and unstructured; in general an algorithm designed to run on an unstructured mesh can also be applied to a structured mesh, but not visa-versa.

In a structured-mesh solver, the computational domain is divided up through curvilinear grid lines, which allow mesh clustering and relaxation along grid lines. This results in a mesh that is composed of structured rows, columns, and stacks of cells. Each cell is indexed by a set of integers, i.e., (i, j, k) in three-dimensional space. The indices i, j, and k represent the number of the mesh elements along the curvilinear coordinates (ξ, η, ζ), which through coordinate transformation corresponds to the Cartesian coordinates (x, y, z). The neighboring cells would have indices $(i \pm 1, j, k)$, $(i, j \pm 1, k)$, and $(i, j, k \pm 1)$, with respect to the central cell at (i, j, k) in the ξ, η, and ζ directions, respectively. This constraint requires that the two-dimensional computational domain is composed of quadrilaterals arranged into rows with the same number of columns in each. In three-dimensional space, the domain is composed of hexahedra where each row of hexahedra have the same number of columns and each stack has the same number of rows. In contrast, an unstructured mesh has no such constraint for cell connectivity. The conventional approach to unstructured mesh discretization is to use triangular cells in two-dimensions and tetrahedrals in three-dimensions. It is also possible to use other cells, such as quadrilaterals for two-dimensional space and hexahedra, pyramid, and prism for three-dimensional space. Moreover, mixed types of cells could also be used in a composed mesh.

Moreover, a common practice in structured-meshed solvers is the use of coordinate transformation between the Cartesian coordinates and the curvilinear coordinates. As such, the finite-volume method employed is imposed over a cube in the transformed space as defined by the transformed coordinates (ξ, η, ζ) for the space-time flux conservation. This transformation can degrade the accuracy of the solver especially in regions where the transformation Jacobian varies greatly from cell to cell. The caveat of this approach is that the mesh metrics are usually calculated by a second-order method, which is incompatible with the high-order algorithms employed for flux conservation[6]. As such the accuracy of the numerical solution could deteriorate.

When a structured mesh is applied to complex geometry, the mesh generator needs to invoke multi-block topology in conjunction with curvilinear coordinates inside each block. Across the neighboring blocks, the mesh cells need to be perfectly aligned for seamless calculation for enforcing flux conservation. This process of generating a mesh across multiple blocks can be rather laborious. In contrast, an unstructured-mesh generator uses an advancing-front algorithm to generate the mesh. These schemes are designed to cover the entire computational space without any internal boundaries between jointed blocks. To accomplish this the algorithm creates cells along the boundaries of the computational domain, then the algorithm works its way into the interior of the spatial domain filling the space behind the advancing front with unstructured cells. These features makes unstructured meshes ideal when meshing complex geometries.

Another advantage of unstructured meshes is the ease at which adaptive mesh refinement is applied. Unstructured-mesh solvers can increase the mesh resolution by subdividing a single cell into multiple cells. If required, a region can be re-meshed to obtain the desired resolution. In contrast, implementing adaptive grid refinement on a structured-meshed solver requires special treatments for calculating fluxes passing the interface between a coarse mesh and a fine mesh. Across the interface, a cell on the coarse-mesh region could have multiple neighbors on the fine-mesh region. The process of calculating the fluxes across the mesh interface could involve modification of the CFD algorithm employed and thus could impact the overall accuracy of the numerical solution.

To recapitulate, unstructured-meshed solvers have many advantages over structured-meshes solvers because (i) the ability and efficiency to mesh a spatial domain with complex geometry, (ii) the easiness in adopting an adaptive-mesh-refinement algorithm on top of the CFD method employed, and (iii) the easiness in achieving high mesh resolution around high-gradient areas such as curved boundaries [7].

The second identifiable characteristic is the order of the method used to solve the PDEs. In general, low-order methods are numerical inexpensive, straight forward to implement, and can more easily handle large gradients. In this case expensiveness refers to the amount of memory and CPU time required to integrate the solution at one cell for one time step. Conversely, high-order methods are more complex and expensive to run as when compared to low-order methods. Although more expensive, high-order schemes can potentially make up for the additional demands by achieving greater accuracy with fewer cells. For example if the error of the numerical results, say in the L_2 norm, was plotted against the computational time, one could find that for the same computational time the numerical solution obtained by a high-order scheme could be more accurate than that of a low-order method. In other words, a high-fidelity solution is more efficiently obtained by increasing the order of the solver rather than increasing the mesh resolution.

Due to the complexity of high-order schemes, the majority of the high-order solvers have been been deployed on structured meshes. When using a structured mesh, the mesh stencil is aligned along curvilinear coordinates which greatly simplifies the computational logic. For example if a more accurate value of $\partial u/\partial \xi$ is required one simply uses a larger stencil of cells along the ξ-axis. Although increasing the stencil size works well on structured meshes it presents several difficulties when applied to an unstructured mesh. This difficulty will be explained in detail in Section 1.2.1.

1.2 High-Order Unstructured-Meshed Schemes

This section reviews several high-order unstructured-meshed CFD methods. The schemes that will be taken into consideration are: (i) the K-exact method[8, 9], (ii) the Spectral Volume (SV) and Spectral Difference (SD) methods[6, 10], and (iii) the Discontinuous Galerkin (DG) method[11–13]. These high-order schemes have demonstrated higher-order convergence on unstructured meshes. The following discussions will focus on the strengths and weakness of each method in the following aspects: (i) the complexity level in the reconstruction procedure, and (ii) the numerical stability constraint.

The K-Exact methods use a larger stencil consisting of many cells, including the central cell where the unknowns are being updated, its adjoining cells, and the surrounding cells, and so on. A drawback of the method is that the choice of the stencil used by the numerical algorithm is not unique and depend on the quality of the mesh. Different stencil choices could lead to different solutions and thus different numerical accuracies.

The SV method achieves higher-order accuracy by decomposing the individual spectral volume into smaller sub cells, in which the conserved variables are discretized and calculated. The SD method differs from the SV method in two aspects: (i) the SD method solves the finite-difference counterpart of the governing equation in the differential form instead of the integral form as that in SV method, and (ii) the SD method has a special reconstruction procedure, in which the higher derivatives of the conserved variables are calculated by using the values of the variables evaluated at the quadrature points of each cell. In the following sections, further comments on each of the above-mentioned high-order method are provided.

The DG method uses multiple degrees of freedom and a higher-order basis functions inside of each cell. This allows a higher-order discretization of the unknowns inside one cell without involving many mesh cells in an extended mesh stencil. As such, only the cells adjoining the central cell, where the unknowns are being updated to the next time step, are used in algorithm. Due to the higher degree of freedom and the use of high-order basis functions, the DG algorithm usually involves solving a linear system of algebraic equations.

1.2.1 The K-Exact and Unstructured WENO Methods

The K-exact method started as a structured solver that was modified for use on unstructured meshes. As a result, the mesh stencil employed is sprawling, containing many more cells beyond the neighboring cells spreading out in a cone like shape. Since this method is performed on unstructured meshes, the calculation of the derivatives of unknowns cannot be performed by finite differencing the unknowns along the mesh stencil grid lines. For example, a spatial derivative of an unknown in the η direction must be calculated by taking into account the values of the cells that that do not share the same values of ξ and ζ coordinates. Moreover, according to the paradigm of the modern upwind method, the one-dimensional Riemann solver has been used as the building block of the numerical scheme. For multiple-dimensional space, a directional splitting approach has been commonly employed. In general, the calculated of the one-dimensional Riemann solver would be aligned with one of the curvilinear grid line. For a two- or three-dimensional problem, the method individually applies the one-dimensional solver in the ξ, η, and/or ζ directions with the understanding that the actual three-dimensional Riemann problem could be approximated by adding two or three one-dimensional Riemann solutions. As a result, an error would be included in the numerical solutions. The amount of the error depends on the topology the mesh employed. It has been impossible to justify this ad hoc approach. To date, it has been considered as the most critical issue of the modern upwind methods.

Barth and Frederickson[8] developed the first high-order K-Exact method for solving fluid mechanics equations. Their scheme used a Finite-Volume (FV) method based on the Godunov's method. The method uses a large stencil, which contains as many cells as the number of the unknowns per governing equation. As mentioned above, there is not a unique way to select the neighboring cells to be included in the algorithm. In other words, it is unclear which neighboring cell should or should not be included in the algorithm. Barth noted that this scheme's reconstruction matrix is ill-conditioned when the aspect ratio of the cells is too large or when a quadratic or higher order interpolation is employed during the reconstruction procedure[9]. One approach to circumvent this problem is to use more cells than are required. This results in an overdetermined system of algebraic equations. To solve this system a least-squares minimization approach is employed to determine the contribution of each cell in the numerical algorithm. As such, one can always have a solution by inverting the matrix.

Later, Durlofsky et al.[14] applied the Essentially Non-Oscillatory (ENO) reconstruction procedure as an integral part in building up a K-Exact method. The reconstruction procedure used by the ENO method is based on adaptively selecting the cells used in the stencil based on the principle of achieving the smoothest reconstruction. By using a different basis function Abgrall was able to form a system with better stability characteristics when compared to previous K-exact methods[15]. Ollivier-Gooch [16] applied the Weighted Essentially Non-oscillatory (WENO) scheme for an extended K-exact method for constructing an unstructured-meshed solver. A large mesh stencil including many nodes are involved in the algorithm. Then, the contribution of each node are weighed for the overall optimal solution. Friedrich reported that the WENO method was no more computational expensive than that of the ENO scheme, but in general the WENO method provides a more accurate solution[17]. He also reported that the ENO scheme could be further optimized if a more efficient stencil selection could be developed.

Dumbser and Kaser[18] reported the first application of the WENO scheme to three-dimensional CFD simulation on unstructured mesh. They demonstrated the results by using a seventh-order WENO method for two-dimensional problems, and a sixth-order method for three-dimensional flows. Dumbser et al.[19] expanded their solver for solving non-fluid problems. This method also employed a quadrature-free integration procedure, where the spatial integral is evaluated analytically instead of a Gauss-quadrature procedure. The implicit temporal integration is done by using the method of Arbitrary high-order schemes using DERivatives (ADER), which combines a one-step time integration with the use of an approximate Riemann solver. The method was originally developed by Toro et al. [20] for structured-meshed solvers based on the upwind method.

A key advantage of the K-Exact schemes when compared to many of the other high-order, unstructured-meshed solvers is that it remains stable as the CFL number approaches one. For example, Ollivier-Gooch ran the simulations with a CFL number of about 0.8[16].

In general, all of the above-mentioned methods suffered from an ill-conditioned reconstruction matrix. The standard remedy was to use a mesh stencil that was larger than required. This resulted in an over-determined system which was solved using a least-squares approach. The requirement of a sprawling mesh stencil is quite cumbersome. For example, the number of cells required for a three-dimensional third-order solver could be 50 to 70 cells, when the solver is increased to fourth-order the number of cells involved in the stencil was estimated to be 120[21, 22]. This particular shortcoming renders the K-Exact method inefficient when running on parallel computers based on domain decomposition.

1.2.2 The Spectral Methods

Both SD and SV methods hold multiple Degrees Of Freedom (DOF) inside each cell. This allows the scheme to use a compact mesh stencil, which contains a central cell where the solution is being sought and its adjoining cells. Another advantage of the SV and SD methods, as compared to the K-Exact methods, is that the reconstruction matrix is never singular[10].

However, both SV and SD schemes suffer a from more stringent CFL constraint for stable calculation as compared to the K-exact methods. In the literature, information about the CFL numbers employed in the reported simulations by the spectral methods have been scant. Liu et al. [2] reported that a CFL number approximately equal to 0.089 was used for one of the two-dimensional, second-order simulations. For other cases, the time increment instead of the CFL number was reported. From this information, the change of the time increment for the other simulations were estimated. The time increment used in the two-dimensional, third-order scheme was decreased to about 58% of that for second-order scheme for the same calculation. For the three-dimensional simulations, the time increment needs to be about 58% and 23% of the two-dimensional simulations when a second- and third-order schemes where used, respectively. Assuming that the wave speed and characteristic length remain constant, the CFL number used for the three-dimensional, third-order simulations would be about 0.02.

A stability analysis for the one-, two-, and three-dimensional solvers has been carried out by Van Den Abeele et al.[23–25]. In the one-dimensional case, weak instabilities were found when the Gauss-Lobatto distribution was used in conjunction with the upwind method for calculating flux[23]. Van Den Abeele concluded that the dispersion and dissipation properties of the numerical solution are better metrics for determining the quality of the scheme than the Lebesgue number. This study was later extended to two-dimensional problems [24]. Again, weak instabilities of the scheme were found and stringent CFL constraints must be heeded when the Lebesgue criteria were used to generate the control volume. Based on the results of this analysis, new third- and fourth-order methods for two-dimensional simulations were proposed. The new methods were found to be more accurate and a larger time step could be used in the calculation. It was reported that the CFL number could be increased from 0.04 to 0.09, when the new fourth-order scheme was used. Van De Abeele et al. applied the same stability analysis to their three-dimensional, unstructured tetrahedron meshed solver and found that their second-order method is be stable as long as the CFL number is well within the constraint. However, they could not find a stable third-order scheme [25].

There has been significant research to make the SV methods more computationally efficient. One such effort was initiated by Harris et al. where they developed a quadrature-free SV method[26]. They were able to reduce the operational count of the method by approximately 72% for both third- and fourth-order schemes. The quadrature free method was extended to three-dimensions by the efforts of Yang et al.[27]. In their study they reported that fewer flux calculations were needed when compared to similar methods using quadratures. However, no information about CPU speed-up was provided. Another way to make the scheme more efficient is to use an implicit time integration procedure. One such work was performed by Breviglieri et al. wherein they where able to use a very large CFL number for steady state problems[28]. A downside to these implicit schemes is that they do not always provide a time accurate solution.

In an effort to provide a simpler, more efficient numerical algorithm Liu, et al. developed the SD method. [6]. Wang et al.[29] extended the SD method to solve the Euler equations. In one test case, they found that a third-order SD method is about 25% faster than the SV method. Stability analysis of the SD method was performed by Van de Abeele et al.[30]. They found that, in general, the position of the solution points does not affect the accuracy or stability of the scheme. This finding simplifies the design of the scheme and reduce the computational cost. They also found that existing schemes that use the Chebyshev-Gauss-Lobatto nodes as the flux points leads to weakened numerical instability. They provided the locations for flux points in one-dimensional mesh as well as in the two-dimensional quadrilaterals, which would ensure stable calculation. However, they could not provide a suitable mesh stencil for two-dimensional triangular meshes which could achieve high-order accuracy with stable calculation.

In summary, the spectral methods has certain advantages over the K-Exact method mainly because its reconstruction matrix is not ill-conditioned. The scheme also has a compact stencil, which allows efficient treatment of mesh nodes along domain-partition line for parallel computation. However, the spectral methods have some drawbacks. The most serious one is the stringent and seemingly uncontrollable CFL stability criteria which limits the maximum usable time step.

1.2.3 The Discontinuous Galerkin Method

The DG method is more mature when compared to the two previous high-order, unstructured-mesh CFD methods presented thus far. The DG scheme was originally developed by Reed and Hill[31] to solve the neutron transport equation. Later, in a series of papers, the DG method was adapted to solve the Euler equations for compressible flows by Cockburn, Shu, and their collaborators[11–13, 32]. The Runge-Kutta (RK) method is used in integrating the equations in time. These initial papers laid the ground work for the Runge-Kutta Discontinuous Galerkin (RKDG) schemes for solving one- and two-dimensional Euler equations. Cockburn and Shu [1] noted that the CFL condition for the RKDG scheme is $\text{CFL} = 1/(2k+1)$, where k is the order of the polynomial employed in the discretization. This equation is exact for $k < 2$. If $k \geq 2$, the estimated CFL is off by about 5%. They also tabulated the CFL constraints as a function of both the order of the polynomial for spatial discretization and the order of the RK scheme for integration in time. They found that as the order of the RK scheme increases, the CFL constraint would relax. The DG scheme has been also been developed to solve the Navier-Stokes equations[33, 34].

An extensive stability study of the RKDG schemes for two-dimensional problems was performed by Toulorge and Desmet[35]. The study used a von Neumann-like procedure to analyze the discrete equations. Their work was done on a structured mesh composed of triangular cells. They discovered that the CFL constraints became tighter as the quality of the grid decreased. The quality of the grid was determined by the ratio of the radius of the inscribed circle to the radius of the circumferential, or outer, circle. This study used two different flux calculations: (i) the Lax-Friedrichs method, and (ii) the upwind method. They found that the maximum stable CFL number for the fourth-order solver ranges from 0.176 to 0.262 and 0.160 to 0.319 for the Lax-Friedrichs method and the upwind method, respectively.

Attempts have been made to modify the RKDG scheme to improve the computational efficiency. In the original DG method, the spatial integration was done using Gauss quadrature, which requires $2N$ floating point operations for N equations at each quadrature point. Atkins and Shu [36] implemented a quadrature-free formulation, and the operation count was reduced to about a quarter of that by using the Gauss quadrature method. However, this new more efficient method has a more restrictive stability criteria than the quadrature method. The quadrature free method requires a CFL number that is between 50% and 10% of the that used by the original Gauss quadrature method.

Another DG method developed by Dumbser and Munz uses the ADER algorithm to solve the Riemann problem[37]. The resulting scheme is a “one-step” method in the sense that the temporal integration is done analytically in one step rather than multiple steps as that in a RK scheme. Using this formulation they achieved eighth-order accuracy when solving the two-dimensional Euler equations. The scheme had a slightly more stringent CFL constraint than that in the RKDG method. For one-dimensional problems, the CFL number was equal to about 90% of that of the RKDG method. For the two-dimensional problems, the CFL number needs to be reduced to be about 45% of that of the RKDG method.

Dumbser et al.[38] extended the ADER-DG scheme to three-dimensional problems. The study reported a refined CFL constraint: CFL $= \alpha/(2k+1)$, where k is the order of the polynomial and $\alpha \leq 1$. When the order of the polynomial is less than 5, $\alpha = 0.7$. For the fifth- and sixth-order methods, $\alpha = 0.5$. Although the CFL constraint is tighter in the ADER-DG scheme than that the original DG method, the ADER-DG method is computationally more efficient due to a more efficient integration algorithm. The ADER-DG scheme is also more efficient when running on a computer cluster for parallel computing because it requires less message to be passed among the computer nodes during the time integration.

Another attempt to speed up the DG method is the p-multi-grid method. The p-multi-grid method was originally developed by Ronquist and Patera for the spectral element method[39]. What makes this procedure unique is that during a single time step multiple time integration procedures are used. Luo, et al. adapted this method for the DG scheme and reported a speed up of 10 times when compared to their second- and third-order explicit solvers[7]. By including an implicit time integration step they where able to use a CFL of 5000 and when using a fully explicit time stepping procedure they achieved a maximum CFL number of 2. However, a draw back of this method is that it has only been used for steady-state solutions.

The DG method and the spectral SV/SD methods have similar strengths. Both methods are more stable and have a more compact stencil than the K-Exact scheme. Several researchers have compared the DG methods with the spectral methods. Zhang and Shu[40] used the one-dimensional convection equation to compared the DG method with the SV method. They found that the numerical results obtained by using the SV method had larger errors, as much as 416% more than that by using the DG method. However, the SV method remained stable at higher CFL numbers.

Sun and Wang [41] reported a detailed comparison between the DG and SV methods in terms of the operational counts, the required computer memory, and the errors of the numerical solution measured in norms. They found that the SV method requires fewer operations and less memory than that of the DG scheme. For a two-dimensional fourth-order scheme, the required floating point operations to march one time step at one cell is 1496 for a scalar convection equation and 7334 for the non-linear Euler equations. The SV scheme requires 1287 and 6168 operations for solving the scalar convection equation and the Euler equations, respectively. For memory, the DG scheme requires 512 words per cell for solving the two-dimensional Euler equations. The SV requires 361 words. For most cases the SV method was faster in computation than the DG method. The one exception was the fourth-order Euler solver. For numerical accuracy, the DG method had consistently smaller error norms than that of the SV method. In these comparisons, the same time step was used for the DG and SV methods, even though a larger time step could have been used for the SV method. When applied to simulate a flow field with a double-Mach reflection, they reported that the SV method was able to provide a more accurate solution of the shock waves than the DG method.

To recapitulate, several high-order methods for unstructured-mesh solvers have been reviewed in the proceeding sections. The K-Exact method is able to use a larger time step but it also uses a large stencil to obtain the higher derivatives of the unknowns in the discretization process. Thus, the K-Exact method is unattractive for HPC or parallel computation. An even bigger problem with the K-Exact method is its ill-conditioned coefficient matrix as a result of the reconstruction procedure, which would lead to various difficulties and complex remedies in solving the linear algebraic equations. The DG and spectral SV/SD methods, on the other hand, use a compact mesh stencil which makes these methods more amenable to parallel computation. When comparing the SV method to the DG method, the SV method requires fewer floating point operations and less computer memory. Moreover, the SV/SD methods have better shock capturing capabilities as compared to the DG method. However, in general, the DG methods are more stable and accurate than the spectral methods. The DG methods also seem to be more widely used than the spectral methods.

1.3 The Space-Time CESE Method

The main contribution of the present dissertation is the successful extension of the space-time CESE method from low-order to high-order, in two- and three-dimensions. And from a one-dimensional fourth order scheme to an arbitrary order accurate scheme in one-dimension. This section provides the background information about the space-time CESE method, which was originally developed by Chang in the early 1990s[4, 5]. In contrast to conventional CFD methods, the CESE method has several unique features. The most significant attribute is the use of a unique space-time integration equation for the conservation laws, different than the Reynolds transport theorem. In fact, the Reynolds transport theorem is a special case of the more general space-time integral in the CESE method. When deriving the space-time integral equation, the governing equations, e.g. Euler Equations, are first cast into a space-time divergence-free formulation. Then, by using the Gauss divergence theorem, defined in the space-time domain, the differential equations in the space-time divergence form is recast into a space-time integral form. This equation is different from and is more general than the traditional Reynolds transport theorem. The CESE method integrates the space-time integral equation to progress solution in time.

To perform the integration, the space-time domain is divided into non-overlapping

Conservation Elements (CEs). The space-time flux conservation is enforced locally over each CE and globally over the whole space-time domain of interest. To facilitate the integration over each CE, Solution Elements (SEs) are define. In a SE the flow variables and the fluxes are discretized and represented by a Taylor series, expanded in both space and time. The order the Taylor series and proper integration over the CE determines the order of the accuracy of the CESE method. In the original CESE method a first-order Taylor series was used and resulted in a second-order accurate solver. In this work, a higher-order, i.e., third-order and beyond, Taylor series will be employed for discretization to yield the high-order CESE methods.

In the time-marching calculation, the Taylor series coefficients of the flow variables and fluxes in each cell at the new time step are sought. As it will be shown in the dissertation, the fluxes, the spatial and temporal derivatives of fluxes, and the temporal derivatives of the flow variables are calculated based on the values of the flow variables and its spatial derivatives. In other words, the primary unknowns of the CESE methods are the flow variables and its spatial derivatives only. Therefore, the operational counts and the required computer memory of the CESE method depend on the number of the flow variables and their spatial derivatives only. This particular feature of the CESE method will be preserved in the high-order extension of the method.

The mesh stencil of the CESE method is similar to that of the Leapfrog and the DuFort-Frankel method, in that the solution marches forward in a time-staggered mesh. However, unlike the leap-frog scheme or the DuFort-Frankel scheme, the CESE method is a two-level scheme, in which the method only requires the solution at the immediately previous time step. In contrast, the solutions at the previous two time steps are required for the leapfrog and the DuFort-Frankel method. Moreover, the staggered space-time marching mesh stencil allows for a unique value of the flux functions at an interface of the conjoined SEs to be determined. This property negates the need of a Riemann solver to determine the values along a face when calculating the flux.

The original second-order CESE method as well as the high-order CESE methods are stable as long as CFL number is ≤ 1. This CFL constraint for stable calculation has been shown theoretically by Chang [3, 4] based on the von Neumann stability analysis of the CESE method for solving both linear and the non-linear Burgers equations. In the past, the theoretical result of the CFL constraint has been thoroughly verified through a multitude of simulations of hyperbolic problems including many non-fluid applications. As will be shown in this dissertation, the same CFL≤ 1 criterion for stable calculation holds up well for the fourth-order CESE method, when applied simulations of both smooth compressible flows and flows with shocks.

The original second-order CESE method has been previously applied to solve a wide varieties of linear and nonlinear hyperbolic problems, including (i) compressible flows modeled by the Euler and the Navier-Stokes equations for high-speed aerodynamics with shocks as well as low-Mach-number, nearly incompressible flows

[5, 42–45]; (ii) combustion problems with multiple species and finite rate kinetics [46–48]; (iii) aero and hydro acoustics with linear and nonlinear waves in the fluid flows [49, 50]; (iv) plasma dynamics modeled by the ideal Magnetic-Hydro-Dynamics (MHD) equations [51, 52]; (v) two-phase water flows with cavitations or air bubbles [53, 54]; (vi) linear and nonlinear waves in complex solids, including anisotropic elastic solids, viscoelastic soft tissues, and plastic solids[55–57]; (vii) the Maxwell equations for electromagnetism[58]; and (viii) acoustic wave shock interactions[50].

1.3.1 Evolution of the CESE Method

Since its inception the methodology of the space-time integration procedure used by the CESE method has remained intact. In the setting of the original second-order CESE method, this space-time integration procedure has been extended from solving the conservation laws in one spatial dimension to problems in two and three spatial dimensions. On the other hand, the central differencing procedure used to determine the odd derivatives of the flow variables, e.g., the first spatial derivatives of the primary unknowns, has seen systematic improvements.

The first such enhancement was the development of the CFL insensitive scheme. Early on, it was discovered that the $a-\epsilon$ scheme, originally developed for one-[5] and two-dimensional[59] CESE solvers, became diffusive as the CFL number approached zero. To mitigate the numerical dissipation, Chang developed the $c-\tau$ scheme [60, 61].

The next improvement to the central differencing procedure in the CESE method was the development of the Edge Based Derivatives (EBD) method by C.-L. Chang [44]. The motivation of the EBD method was that the $c - \tau$ scheme produced poor results when the aspect ratio of a cell was greater than approximately 100. Meshes composed of cells with large aspect ratios are commonplace when using the Navier-Stokes equations to simulate viscous flows. For example in the near-wall regions, slender cells are used to resolve the boundary layer. By using the EBD scheme, meshes composed of skewed cells with the aspect ratios as large as 10^6 are possible.

Another import improvement to the CESE scheme was the development of a local time stepping procedure by Chang [62–64]. This scheme allows different cells to take different time steps. Variations in time increments among the cells in a mesh could be of 2 to the k-th power. For example, if the maximum time step in the computational domain is $\Delta t_{\max}$, other local time steps could be equal to $\Delta t_{\max}/(2^k)$, where $k = 0, 1, 2, \ldots$. This would allow each cell to have a local CFL number close to a desired value, e.g. unity. Moreover, as shown by Yen[62] and Chang[63], the local time stepping procedure can also be used for efficient calculation of time-accurate solutions.

Finally, it is worthy to note that the space-time integral equation employed by the CESE method is inherently suitable for simulations with moving meshes, such as fluid-solid interactions. Such capabilities have been developed by Cook et al.[65], wherein the CESE fluid solver is coupled with a Finite Element Analysis (FEA) solver for solid mechanics to simulate the motion and deformation of the solid medium induced by fluid motion around the solid.

To date, a key criticism against the CESE method was that the method lacks a high-order extension. Previous attempts to extend the scheme to be high-order in accuracy resulted in more stringent CFL constraints and undesirable complexity in the algorithm[66]. This changed in 2010 when Chang[3] reported a new one-dimensional, fourth-order CESE scheme which remained stable when the CFL equaled one. This was shown to be true for both the linear convection equation as well as the non-linear burgers equation[3]. This work is important because it proved that the high-order CESE scheme maintains the same stability constraint, i.e. CFL number ≤ 1. Furthermore, it showed that the fundamental structure of the space-time integration in the high-order CESE method remains the same as the original second-order CESE method. This means that the same compact mesh stencil used in the original method is used in the new high-order CESE method. Essentially, all advantageous features of the original CESE method have been retained in the new scheme. This paper also provided a sketch for a multidimensional system, making an important observation on the ability to use the alternate rule of differentiation.

1.4 Objectives of the Dissertation Work

In this chapter, a review on the existing high-order, unstructured-meshed CFD solvers based on the K-Exact method, the DG method, and the spectral methods was shown in Section 1.2. Suffice it to say that there is room for significant improvement in accuracy and robustness for high-order, unstructured-meshed CFD solver. On the other hand, the second-order CESE method has great potential to become a mainstream numerical framework with a long life cycle for time accurate solutions of all sorts of hyperbolic problems including the aerodynamics equations.

By using the original second-order CESE method for one-, two-, and three-dimensions in conjunction with the methodology found in the newly developed fourth-order extension of the CESE method the scope of the present dissertation is to expand the CESE scheme to high-order for time-accuracy solutions of hyperbolic problems and conservation laws in in one-, two-, and three-dimensional space. Furthermore, it is desired that these new algorithms retain the favorable attributes of the original second-order CESE scheme. Specifically, the envisioned high-order CESE method and the associated CFD codes must fulfill the following objectives:

1. The new high-order method represents a framework for a hierarchy of schemes, which allow systematic extension to higher order methods, i.e., fourth-, sixth-, eighth-order and beyond.

2. The new high-order method must be based on integrating the space-time flux conservation equation, instead of the Reynolds transport theorem for the conservation laws. As such, the space-time flux in the time-marching calculation is always conserved.

3. The method must have a compact mesh stencil, which involves only immediate

conjoining mesh cells of the central cell, where the unknowns are being solved in the time-marching calculation.

4. The method must retain the same stability criteria as the second-order CESE method, i.e., a CFL number ≤ 1. Preferably, CFL ≈ 1 can be achieved for all high-order options of the scheme for efficient time-marching calculation.

5. The method must be truly multi-dimensional without relying on directional splitting for calculating fluxes in multi-dimensional space. This allows the integration formulation developed for one-dimensional problems to be straightforwardly extended to two-, and tree-dimensional problems without losing any accuracy in the formula.

6. The method must be able to handle complex geometries through unstructured mesh composed of mixed elements, including the use of tetrahedrons, hexagons, pyramids, and prisms for a three-dimensional problem.

To achieve the above objectives, a high-order space-time CESE method and codes have been developed for solving two- and three-dimensional hyperbolic PDEs. To demonstrate the capability of the new solver, I will focus on solving the Euler equations for compressible flows with and without shock waves. The work reported here involves extensive works in three aspects: algorithm development, code development, and code validation. The overall goal is to deliver a robust, highly accurate, two- and three-dimensional, high-order CESE code for use on unstructured meshes. For code validation, standard benchmark test cases, in one-, two-, and three-dimensions where be performed and the results where compared to either: theoretical solutions, previously reported CFD results, or experimental data.

The code development effort including the debugging process of a unstructured-meshed CFD code is a daunting and complicated task. To mitigate the computer-science aspect of this endeavor, an in-house, open-sourced numerical framework, SOLver CONstructor (SOLVCON)[67], has been adopted. A framework is a piece of software that provides a platform to aid in the development of future software. This is accomplished by providing basic functionality found across software of a given type and provides a non-intrusive way to insert custom numerical algorithms. In particular SOLVCON provides input/output (IO) logic, domain decomposition, message passing, and the ability to insert custom routines.

The remainder of this dissertation is divided into the following five chapters: Chapter 2 presents the derivation and applications of the one-dimensional high-order CESE method. Chapter 3 provides the derivation of the two-dimensional, fourth-order CESE method. This chapter also includes code validation, including two-dimensional numerical results of various test cases obtained by using the newly developed high-order CESE code. Chapter 4 illustrates the derivation of the three-dimensional fourth-order CESE method. The chapter also reports numerical results of several test cases for code validation. Chapter 5 presents the numerical implementation of the new solver for hyperbolic problems in two- and three-dimensional spaces.

Chapter 6 provides a summary of this research work and recommends possible future work in certain topics.

CHAPTER 2

THE ONE-DIMENSIONAL, HIGH-ORDER CESE METHOD

In this chapter, the high-order CESE method for solving a set of one-dimensional coupled, linear or nonlinear, first order hyperbolic Partial Differential Equations (PDEs) is discussed. In particular, the Euler equations for compressible flow will be used as an example. Nevertheless, the derivation here is general and can be applied to various other hyperbolic problems and conservation laws. Moreover, a generic formulation for nth-order CESE will be presented.

To proceed, a set of one-dimensional PDEs cast into a vector form are considered:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{S}, \tag{2.1}$$

where $\mathbf{U}, \mathbf{F}$, and $\mathbf{S}$ are the column vectors for the primary unknowns, the flux functions, and the source terms, respectively:

$$\begin{aligned}\mathbf{U} &\equiv (u_1, u_2, u_3, \cdots, u_{N_{eq}})^t,\\ \mathbf{F} &\equiv (f_1, f_2, f_3, \cdots, f_{N_{eq}})^t,\\ \mathbf{S} &\equiv (s_1, s_2, s_3, \cdots, s_{N_{eq}})^t,\end{aligned}$$

where u_k, f_k, and s_k with $k = 1, 2, \ldots, N_{eq}$ are the entries of the corresponding vectors. N_{eq} is the number of equations. The superscript t denotes transpose. Equation (2.1) can be cast into the following divergence equations:

$$\nabla \cdot \mathbf{h}_i = s_i \tag{2.2}$$

with $i = 1, 2, \ldots, N_{eq}$, where the space-time flux vector $\mathbf{h}_i$ is defined as

$$\mathbf{h}_i = (f_i, u_i)^t. \tag{2.3}$$

The divergence in Eq. (2.2) operates in the two-dimensional Euclidean space $\mathcal{E}_2$ with (x, t) as the independent coordinates. Aided by the space-time flux vector $\mathbf{h}_i$ and Gauss' divergence theorem, the original PDEs, Eq. (2.1), is transformed into the following more fundamental space-time integral form of the conservation laws:

$$\oint_{S(V)} \mathbf{h}_i(x,t) \cdot d\mathbf{s} = \int_V s_i dV, \tag{2.4}$$

where $S(V)$ is the boundary of a space-time region V, and $d\mathbf{s} = d\sigma\mathbf{n}$. Where $d\sigma$ and $\mathbf{n}$ are respectively the differentiable area and outward facing normal vector of the surface element on $S(V)$. Figure (2.1) shows an arbitrary space-time domain V and its surface $S(V)$, over which the space-time integration by using Eq. (2.4) can be performed. The Left Hand Side (LHS) of Eq. (2.4) is integrated over the region $S(V)$ by the CESE scheme. The Right Hand Side (RHS) of Eq. (2.4) is the integration of the source term over the same space-time domain. The method employed to carry out the space-time integration of the source terms could vary depending on the characteristics of the source term, e.g. linear or non-linear sources with respect to the primary unknowns, stiff or non-stiff, etc. In the CESE method, Eq. (2.4) is integrated over each Conservation Element (CE), which in turn is defined by the mesh stencil of the method.

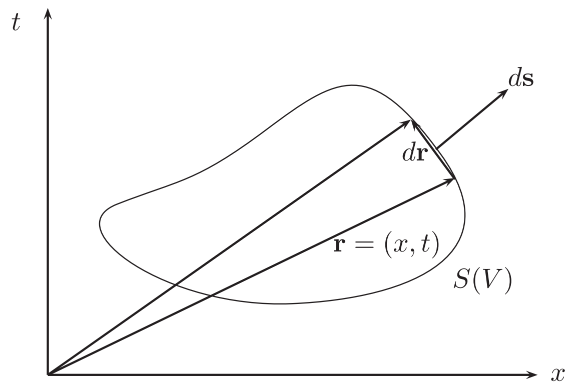


Figure 2.1: An generic closed space-time domain V and its surface $S(V)$, over which the space-time integration, Eq. (2.4), is performed.

2.1 One-Dimensional Discretization

In the CESE method, the space-time domain is divided into non-overlapping CEs, over which the space-time flux conservation is enforced individually over each CE and collectively over the entire domain. On the other hand, SEs are used for discretizing the unknowns and fluxes. The integration is facilitated by the discretized values of the primary unknowns and the flux functions in each Solution Element (SE). The discretized values of the unknowns and the fluxes are used to facilitate the numerical integration. Together, CEs and SEs form a space-time mesh stencil. In deriving the high-order CESE methods, the mesh stencil employed used in this one-dimensional derivation is identical to the one used by Yu[68]. For completeness, the description of the mesh stencil is repeated here.

Figure (2.2) shows one possible space-time mesh configuration. The abscissa denotes the one-dimensional space. In this figure the ordinates denote the time marching direction, the red arrows show the propagation of the solution in the time-marching calculation. As shown in Fig. (2.2), the space-time domain is divided into rhombus-shaped SEs. The same space-time domain is also divided into CEs, over which the space-time integration by using Eq. (2.4) is performed. Figure (2.3) shows three rhombus shaped SE associated with one CE. The rhombus shaped SE is used over the more standard rectangular SE when a source term is present. The rhombus shape of the SE is required when a source term is present in the governing equation. It is not required to modify the definition of the CE, original shown in[5].

Figure (2.4) shows several important locations on the CE. These point are by both the space-time integration and the central differencing procedure. Based on the one-dimensional CESE method. The solid dots A, C, and E are the solution points and are located at (x_j, t^n), $(x_{j-1/2}, t^{n-1/2})$ and $(x_{j+1/2}, t^{n-1/2})$, respectively. Point M^+ is located midway between A and F and M^- is midway between points A and B. Point P^+ is located between point M^+ and point F and point P^- is between M^- and B. The distance between $P^\pm$ and $M^\pm$ is determined by a parameter τ. Associated with the solution point A, the rectangles $ABCD$ and $ADEF$ are the Basic CEs (BCEs). The rectangle $BCEF$, i.e. the union of $ABCD$ and $ADEF$, is called a Compound CE, or CCE.

To proceed, let SE(j, n) denote the SE with its solution point at the mesh node

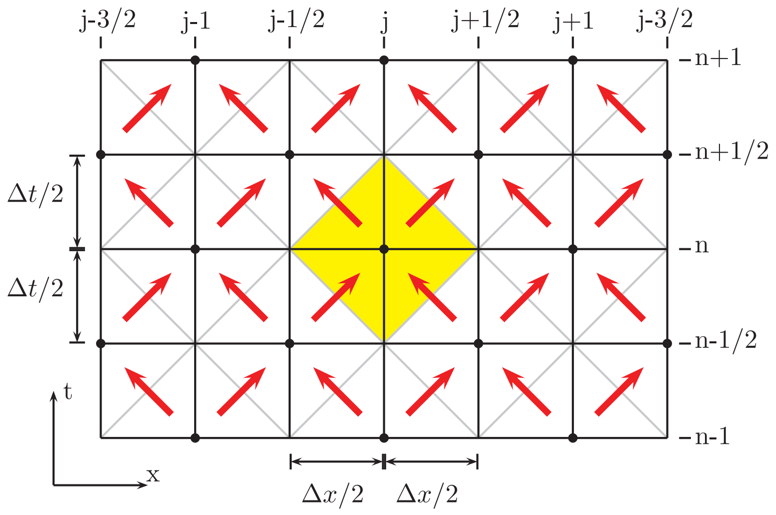


Figure 2.2: The mesh stencil and the zigzagging pattern of the time-marching calculation in the one-dimensional CESE method.

(x_j, t^n). Inside the SE(j, n), the flow variables u_i, $i = 1, \ldots, N_{eq}$ are discretized by a Taylor series. If a fourth-order CESE method is employed than a third-order,

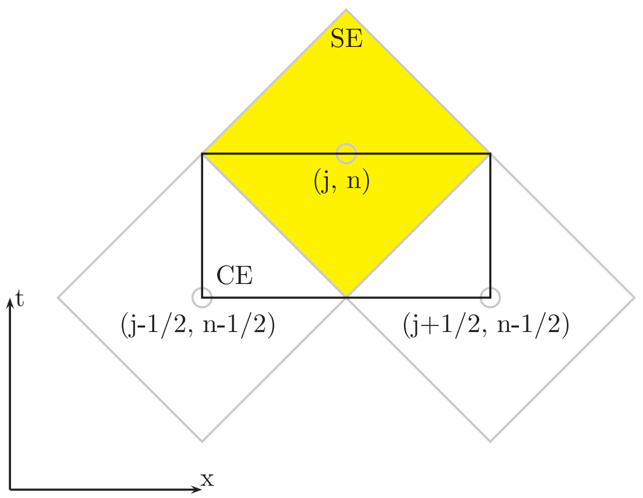


Figure 2.3: The Solution Element (SE) and Conservation Element (CE) of the one-dimensional CESE method.

two-dimensional Taylor series would be used:

$$
\begin{aligned}
u_i^*(x,t;j,n) \equiv\ & (u_i)_j^n + (u_{i,x})_j^n(x-x_j) + (u_{i,t})_j^n(t-t^n) \\
& + \frac{1}{2}\left[(u_{i,xx})_j^n(x-x_j)^2 + (u_{i,tt})_j^n(t-t^n)^2 + \right] + (u_{i,xt})_j^n(x-x_j)(t-t^n) \\
& + \frac{1}{6}\left[(u_{i,xxx})_j^n(x-x_j)^3 + (u_{i,ttt})_j^n(t-t^n)^3\right] \\
& + \frac{1}{2}\left[(u_{i,xxt})_j^n(x-x_j)^2(t-t^n) + (u_{i,xtt})_j^n(x-x_j)(t-t^n)^2\right] + \\
& O(x-x_j)^4 + O(t-t^n)^4.
\end{aligned}
\tag{2.5}
$$

The superscript $*$ in Eq. (2.5) denotes the value of the discretized unknown at a

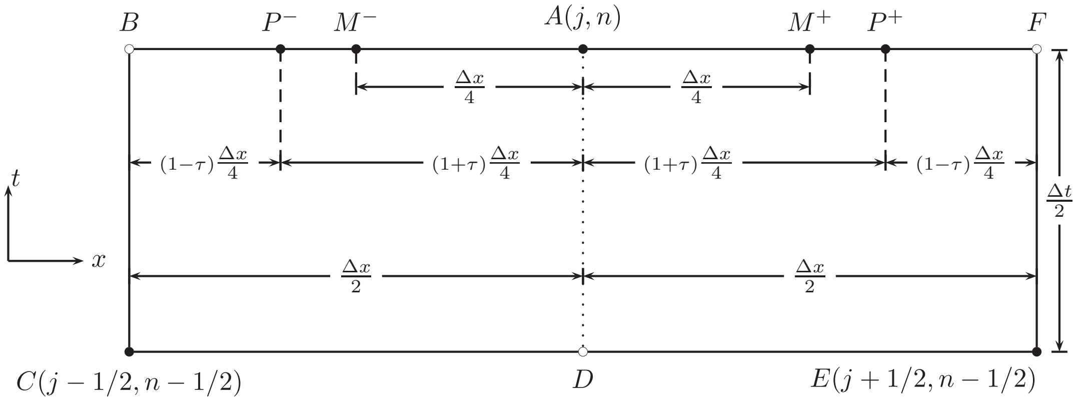


Figure 2.4: The integration of the space-time flux over CEs based on the one-dimensional CESE method.

space-time location (x, t) inside the SE(j, n). In Eq. (2.5),

$$u_{i,x} \equiv \frac{\partial u_i}{\partial x}, \quad u_{i,xx} \equiv \frac{\partial^2 u_i}{\partial x^2}, \quad u_{i,xxx} \equiv \frac{\partial^3 u_i}{\partial x^3}.$$

Similar notations are used for temporal derivatives and space-time cross derivatives of u_i. And, $(u_{i,x})_j^n$ denotes the value of the derivative $u_{i,x}$ at the solution point (x_j, t^n). The same notation is applied to other derivatives.

Equation (2.5) is valid for the fourth-order CESE method. This equation is easily generalized for any higher order system. This generalization is shown in the following equation:

$$u_i^*(x,t;j,n) \equiv \sum_{a=0}^{N_M} \sum_{b=0}^{N_M-a} \frac{(u_{i,x^a t^b})_j^n}{a!b!} \left(x - x_j\right)^a \left(t - t^n\right)^b, \tag{2.6}$$

where

$$u_{i,x^I t^J} \equiv \frac{\partial^{I+J} u_i}{\partial x^I \partial t^J},$$

N_M is the order of the Taylor series used by the CESE method. For example a fourth-order CESE method uses a third-order Taylor series and N_M would equal 3. All coefficients in the Taylor series are defined and held constant at the solution point (x_j, t^n). Similarly, the flux functions f_i, $i = 1, \ldots, N_{eq}$, are discretized by the Taylor series expansion inside the SE(j, n):

$$f_i^*(x, t; j, n) \equiv \sum_{a=0}^{N_M} \sum_{b=0}^{N_M - a} \frac{(f_{i,x^a t^b})_j^n}{a!b!} \left(x - x_j\right)^a \left(t - t^n\right)^b . \tag{2.7}$$

To proceed, the spatial and temporal derivatives of primary variable u_i could also be discretized by using the Taylor series expansion inside a SE. Consider the fourth-order CESE method, then as a result of Eq. (2.5), we have

$$
\begin{aligned}
u^*_{i,x}(x,t;j,n) &= (u_{i,x})^n_j + (u_{i,xx})^n_j(x-x_j) + (u_{i,xt})^n_j(t-t^n) \\
&+ \frac{1}{2}\left[(u_{i,xxx})^n_j(x-x_j)^2 + (u_{i,xtt})^n_j(t-t^n)^2 + \right] \\
&+ (u_{i,xxt})^n_j(x-x_j)(t-t^n) + O(x-x_j)^3 + O(t-t^n)^3, \\
u^*_{i,t}(x,t;j,n) &= (u_{i,t})^n_j + (u_{i,xt})^n_j(x-x_j) + (u_{i,tt})^n_j(t-t^n) \\
&+ \frac{1}{2}\left[(u_{i,xxt})^n_j(x-x_j)^2 + (u_{i,ttt})^n_j(t-t^n)^2 + \right] \\
&+ (u_{i,xtt})^n_j(x-x_j)(t-t^n) + O(x-x_j)^3 + O(t-t^n)^3, \\
u^*_{i,xx}(x,t;j,n) &= (u_{i,xx})^n_j + (u_{i,xxx})^n_j(x-x_j) + (u_{i,xxt})^n_j(t-t^n) + \\
&O(x-x_j)^2 + O(t-t^n)^2, \\
u^*_{i,tt}(x,t;j,n) &= (u_{i,tt})^n_j + (u_{i,xtt})^n_j(x-x_j) + (u_{i,ttt})^n_j(t-t^n) + \\
&O(x-x_j)^2 + O(t-t^n)^2, \\
u^*_{i,xt}(x,t;j,n) &= (u_{i,xt})^n_j + (u_{i,xxt})^n_j(x-x_j) + (u_{i,xtt})^n_j(t-t^n) + \\
&O(x-x_j)^2 + O(t-t^n)^2.
\end{aligned}
\tag{2.8}
$$

The coefficients of the Taylor series for the third-order terms are constant inside the SE(j,n) and are identical to that in the Taylor series for the primary unknowns u_i shown in Eq. (2.5).

In general, the discretized derivatives of u_i with the I-th derivative in space and Jth derivative in time is easily expressed by the following Taylor series expansion anchored at the solution point (x_j, t^n):

$$u^*_{i,x^I t^J}(x,t;j,n) = \sum_{a=0}^{A} \sum_{b=0}^{B-a} \frac{\left(u_{i,x^{a+I}t^{b+J}}\right)_j^n}{a!b!} \left(x - x_j\right)^a \left(t - t^n\right)^b , \tag{2.9}$$

where $A = N_M - I$ and $B = N_M - I - J$. Similarly, the profile of the $(I + J)$th space-time derivative of the flux function f_i inside a CE with the solution point at (x_j, t^n) can also be represented by a Taylor series expansion formulated in terms of higher-order space-time derivatives:

$$f^*_{i,x^I t^J}(x,t;j,n) = \sum_{a=0}^{A} \sum_{b=0}^{B-a} \frac{\left(f_{i,x^{a+I}t^{b+J}}\right)_j^n}{a!b!} \left(x - x_j\right)^a \left(t - t^n\right)^b . \tag{2.10}$$

For the profiles of the primary unknowns and the flux functions to be fully defined inside an SE, all coefficients of the Taylor series for u_i, Eq. (2.9), and f_i, Eq. (2.10), need to be calculated at all solution points in the time-marching scheme. However, it will be shown that not all of the coefficients in the Taylor series are independent. In the following section it will be shown that the only independent, or primary variables are the conserved variables, u_i, and its spatial derivatives, $u_{i,x}$, $u_{i,xx}$, $\ldots$.

2.2 One-Dimensional Independent Variables

In this section it will be shown that the only independent variables in the time-marching calculation are the conserved variables and their spatial derivatives. Other variables, i.e. the temporal derivatives of the conserved variables ($u_{i,t}$, $u_{i,xt}$, $u_{i,tt}$, $\ldots$)

and the fluxes and its spatial and temporal derivatives, can be readily calculated from the primary unknowns $(u_i)_j^n$ and its spatial derivatives.

In the following, a detailed derivation that relates the fluxes and their space-time derivative to the derivatives of the primary unknowns and their derivatives. Two different approaches where employed to accomplish this. The first method, shown below, uses the Jacobian and its derivatives to relate the fluxes back to the primary unknowns. The second method calculates the spatial and temporal derivatives of the fluxes directly through a generalization of Leibniz rule for differentiation. The Jacobian method is shown below because it is more generic and not tied to any particular physics. The Leibniz method was used in the numerical code because it is easier and more efficient to code. This method is further explained in Section 5.1.

Given the model equations, the fluxes are known functions of the primary unknowns. Therefore, the derivatives of the fluxes can be determined through the chain rule. To proceed, let the first-, second-, and third-order Jacobian matrices be denoted by

$$f_{i,l} \equiv \frac{\partial f_i}{\partial u_l}, \quad f_{i,lk} \equiv \frac{\partial^2 f_i}{\partial u_l \partial u_k}, \quad f_{i,lkp} \equiv \frac{\partial^3 f_i}{\partial u_l \partial u_k \partial u_p}, \quad \ldots \tag{2.11}$$

where $i, l, k, p = 1, \ldots, N_{eq}$. Aided by the Jacobian matrices, the space-time derivatives of flux functions can be expressed as functions of u_i with $i = 1, 2 \ldots, N_{eq}$ and their space-time derivatives:

$$\frac{\partial f_i}{\partial \Psi_1} = \sum_{l}^{N_{eq}} f_{i,l} \frac{\partial u_l}{\partial \Psi_1}, \tag{2.12}$$

where $\Psi_1 = \{x, t\}$;

$$\frac{\partial^2 f_i}{\partial \Psi_1 \partial \Psi_2} = \sum_{l}^{N_{eq}} f_{i,l} \frac{\partial^2 u_l}{\partial \Psi_1 \partial \Psi_2} + \sum_{l,k}^{N_{eq}} f_{i,lk} \frac{\partial u_l}{\partial \Psi_1} \frac{\partial u_k}{\partial \Psi_2} \tag{2.13}$$

where $(\Psi_1, \Psi_2) = \big\{(x,x), (t,t), (x,t)\big\}$; and

$$\begin{aligned} \frac{\partial^3 f_i}{\partial \Psi_1 \partial \Psi_2 \partial \Psi_3} =& \sum_{l}^{N_{eq}} f_{i,l} \frac{\partial^3 u_l}{\partial \Psi_1 \partial \Psi_2 \partial \Psi_3} + \\ & \sum_{l,k}^{N_{eq}} f_{i,lk} \left(\frac{\partial^2 u_l}{\partial \Psi_1 \partial \Psi_2} \frac{\partial u_k}{\partial \Psi_3} + \frac{\partial^2 u_l}{\partial \Psi_1 \partial \Psi_3} \frac{\partial u_k}{\partial \Psi_2} + \frac{\partial^2 u_l}{\partial \Psi_2 \partial \Psi_3} \frac{\partial u_k}{\partial \Psi_1} \right) + \\ & \sum_{l,k,p}^{N_{eq}} f_{i,lkp} \frac{\partial u_l}{\partial \Psi_1} \frac{\partial u_k}{\partial \Psi_2} \frac{\partial u_p}{\partial \Psi_3} \end{aligned} \tag{2.14}$$

where $(\Psi_1, \Psi_2, \Psi_3) = \big\{(x,x,x), (t,t,t), (x,x,t), (x,t,t)\big\}$, for $i = 1, \ldots, N_{eq}$. The space-time derivatives of the fluxes that are required for a fourth-order CESE scheme are shown in Eq. (2.14). This method can be extended to orders higher than fourth by taking further derivatives of the fluxes with respect to the conserved variables. A drawback of this approach is that with each additional derivative the flux functions become increasingly complex. This is easily seen in Eq. (2.14). However, when solving a linear hyperbolic PDE the fluxes are a linear function of the conserved variables. This means that all second and greater derivatives of f_i with respect to u_l are null. This greatly simplify the calculation of the derivatives of the fluxes.

To proceed, it will be shown that the derivatives of the primary unknowns u_i involving temporal differentiation can be recast into expressions in terms of the spatial derivatives of the primary unknowns. For the first temporal derivative of the primary unknown, $(u_{i,t})_j^n$, the following algebraic equation is considered

$$(u_{i,t})_j^n = (s_i)_j^n - (f_{i,x})_j^n. \tag{2.15}$$

The equation comes about based on the assumption that the discrete analog of the original PDE calculated at the solution point is valid. In the second-order CESE method, this equality can be rigorously derived based on the space-time flux conservation over a rhombus-shaped CE [5], which coincides with an SE. For the fourth- and sixth-order CESE methods in the present development, however, no such implication can be derived. Suffice it to say that Eq. (2.15) is simply a plausible assumption.

Next, the original model equation, Eq. (2.1), is differentiated with respect to space and time for additional higher-order PDEs and their algebraic analogs at the solution point (x_j, t^n) are:

$$\begin{aligned}
(u_{i,tx})_j^n &= (s_{i,x})_j^n - (f_{i,xx})_j^n, \\
(u_{i,tt})_j^n &= (s_{i,t})_j^n - (f_{i,xt})_j^n, \\
(u_{i,txx})_j^n &= (s_{i,xx})_j^n - (f_{i,xxx})_j^n, \\
(u_{i,ttx})_j^n &= (s_{i,tx})_j^n - (f_{i,xtx})_j^n, \\
(u_{i,ttt})_j^n &= (s_{i,tt})_j^n - (f_{i,xtt})_j^n, \\
&\dots .
\end{aligned}$$

Further derivatives of Eq. (2.1) can be taken to obtain even higher derivatives of $(u_i)_j^n$ as required. Alternatively, this differentiation can be written in a more generic form as:

$$(u_{i,x^I t^J})_j^n = (s_{i,x^I t^{J-1}})_j^n - (f_{i,x^{I+1} t^{J-1}})_j^n \tag{2.16}$$

for all the primary unknowns, $i = 1, 2, \dots, N_{eq}$. The superscripts $I = 0, 1, \dots, N_M$ and $J = 1, 2, \dots, N_M - I$ denote the order of the Taylor series coefficients. As mentioned above, the spatial and temporal derivatives of the flux functions shown in Eq. (2.16) can be obtained through the spatial and temporal derivatives of the primary unknowns u_i in conjunction with the use of the Jacobian matrices and the chain rule.

As a result, the only independent variables in the time-marching calculation are the primary unknowns u_i and their spatial derivatives.

For the fourth-order CESE method, the independent unknowns of each model equation include the following four Taylor series coefficients: $(u_i)_j^n$, $(u_{i,x})_j^n$, $(u_{i,xx})_j^n$, and $(u_{i,xxx})_j^n$. When a fourth-order CESE method is employed to solve the one-dimensional Euler equations, where $N_{eq} = 3$, there will be total of 12 unknowns per cell in the time-marching calculation. In general, if a (N_M+1)th-order CESE method is employed to solve a set of N_{eq}, coupled hyperbolic PDEs, there are $(N_M + 1) \times N_{eq}$ unknowns at each mesh cell. These $(N_M + 1) \times N_{eq}$ unknowns can be categorized into two groups: (i) the even derivatives of the primary unknowns u_i, including the zeroth derivative $(u_i)_j^n$, the second derivative, $(u_{i,xx})_j^n, \ldots,$ and the k-th derivative $(u_{i,x^{2k}})_j^n$, where $k = (N_M - 1)/2$; and (ii) the odd derivatives of u_i, including the first derivative $(u_{i,x})_j^n$, the third derivative $(u_{i,xxx})_j^n$, $\ldots$, and the kth derivative $u_{i,x^{2k+1}}$, where $k = (N_M - 1)/2$. The numerical algorithm applied to calculate the even derivatives of u_i in the time-marching calculation is different from that for solving the odd derivatives of u_i.

The calculation of the even derivatives of the primary unknowns is based on enforcing space-time flux conservation. Which is accomplished by integrating the space-time integral equation defined in the original CESE method[5]. Unlike the conventional finite-volume methods, the flux conservation is not formulated based on the use of the Reynolds transport theorem, which is only a special case of the more general space-time flux integral employed in the CESE method.

For the high-order CESE methods, the space-time flux conservation is enforced based on the original equations as well as the additional space-time integral equations, which must be obtained by first differentiating the original hyperbolic PDEs in space and time and then applying the Gauss divergence theorem to the new PDEs.

On the other hand, the odd derivatives of the primary unknowns u_i at the solution point (x_j, t^n) are calculated based on a central-difference-like procedure. For example, one the primary unknowns $(u_i)_j^n$ are calculated, its first derivatives $(u_{i,x})_j^n$ are calculated through a central differencing procedure. In a similar fashion the third derivatives $(u_{i,xxx})_j^n$ can be calculated by central differencing the second derivatives $(u_{i,xx})_j^n$. In the following 2 sections, a general high-order CESE c-τ scheme for solving a system of one-dimensional N_{eq} hyperbolic PDEs is developed.

2.3 Even Spatial Derivatives of u_i

The key idea in calculating the even spatial derivatives of u_i in the time marching calculation is to have additional equations for the higher derivatives. For example, the following equation will be used as an additional PDE with $u_{i,xx}$ as the unknown:

$$\frac{\partial u_{i,xx}}{\partial t} + \frac{\partial f_{i,xx}}{\partial x} = s_{i,xx},$$

In general, the above equation can be understood as an extension of the original model equation, Eq. (2.1), by differentiating it with respect to x twice. Here, $u_{i,xx}$ is the new primary unknown and the new equation is a first-order PDE. In general, all additional equations for the even spatial derivatives of u_i can be cast into the following form:

$$\frac{\partial u_{i,x^{2I}}}{\partial t} + \frac{\partial f_{i,x^{2I}}}{\partial x} = s_{i,x^{2I}}, \tag{2.17}$$

where $I = 0, 1, \ldots, (N_M - 1)/2$. Aided by the Gauss divergence theorem, Eq. (2.17) is recast into the space-time integral equation:

$$\oint_{S(V)} \mathbf{h}_{i,x^{2I}} \cdot d\mathbf{s} = \int_V s_{i,x^{2I}}, \tag{2.18}$$

where $S(V)$ is the boundary of a space-time region V, and $d\mathbf{s}$ is a differential surface element pointing outward with respect to V. The space-time flux vector $\mathbf{h}_{i,x^{2I}}$ is defined as

$$\mathbf{h}_{i,x^{2I}} \equiv (f_{i,x^{2I}}, u_{i,x^{2I}}).$$

In Eq. (2.18), the Gauss divergence theorem operates in a two-dimensional Euclidean space $\mathcal{E}_2$, in which the independent coordinates are (x, t). The original idea of the space-time integration used in the CESE method can be found in Fig. (2.1). To proceed, the following definitions are made to facilitate the derivation:

$$u^*_{m\bar{x}^I\bar{t}^J} \equiv \frac{\partial^{I+J} u_i^*}{\partial x^I t^J} \left(\frac{\Delta x}{4}\right)^I \left(\frac{\Delta t}{4}\right)^J \tag{2.19}$$

$$f^*_{m\bar{x}^I\bar{t}^J} \equiv \frac{\partial^{I+J} f_i^*}{\partial x^I t^J} \left(\frac{\Delta x}{4}\right)^I \left(\frac{\Delta t}{4}\right)^J \tag{2.20}$$

$$s_{m\bar{x}^I\bar{t}^J} \equiv \frac{\partial^{I+J} s_i}{\partial x^I t^J} \left(\frac{\Delta x}{4}\right)^I \left(\frac{\Delta t}{4}\right)^J \tag{2.21}$$

where $\Delta x = x_{j+1/2} - x_{j-1/2}$ and $\Delta t = t^n - t^{n-1}$. In order to write the equations more compactly, any local constant enclosed within a square bracket will be evaluated at the location specified by the subscript and superscript written on the enclosing square bracket, e.g.:

$$(u_{i,xx})_j^n + (u_{i,xxx})_j^n \frac{\Delta x}{2} + (u_{i,xxt})_j^n \frac{\Delta t}{2} \equiv \left[u_{i,xx} + u_{i,xxx}\frac{\Delta x}{2} + u_{i,xxt}\frac{\Delta t}{2} \right]_j^n .$$

Applying Eq. (2.18) to the CE shown in Fig. (2.4) yields

$$\oint_{S(V)} \left(-u^*_{i,x^{2I}}, f^*_{i,x^{2I}} \right) \cdot (dx, dt) = s^*_{i,x^{2I}}, \tag{2.22}$$

where $I = 0, 1, 2, \ldots, (N_M - 1)/2$.

To find the even derivatives at the new time level Eq. (2.22) is integrated over each line segment in Fig. (2.4). The sign convention used for the integration is counter-clockwise positive. A detailed derivation will only be provided for the first two line segments, AB and BC. For all other just the resulting integration will be provided.

$$
\begin{aligned}
\int_{AB} \left(-u^*_{i,x^Z}, f^*_{i,x^Z}\right) \cdot (dx, dt) &= \int_{x_j}^{x_{j-1/2}} -u^*_{i,x^Z} dx \\
&= -\int_{x_j}^{x_{j-1/2}} \sum_{a=0}^{N_M-Z} \frac{1}{a!} \left(u_{i,x^{a+Z}}\right)_j^n \left(x - x_j\right)^a dx \\
&= -\sum_{a=0}^{N_M-Z} \frac{1}{a!} \left(u_{i,x^{a+Z}}\right)_j^n \int_{x_j}^{x_{j-1/2}} \left(x - x_j\right)^a dx \\
&= \frac{\Delta x}{2} \sum_{a=0}^{N_M-Z} \frac{(-1)^a 2^a}{(a+1)!} \left(u_{i,\bar{x}^{a+Z}}\right)_j^n \left(\frac{4}{\Delta x}\right)^Z
\end{aligned}
$$

where Z is any even derivative, e.g. $Z = 0, 2, 4, 6, \ldots, N_M - 1$. Note that since x is the only dependent variable all of the other terms are removed from the integral. The resulting integration for the BC line segment is:

$$
\begin{aligned}
\int_{BC} \left(-u^*_{i,x^Z}, f^*_{i,x^Z}\right) \cdot (dx, dt) &= -\int_{t_n-1/2}^{t_n} f^*_{i,x^Z} dt \\
&= -\int_{t_n-1/2}^{t_n} \sum_{a=0}^{N_M-Z} \frac{1}{a!} \left(f_{i,x^Z t^{Z+a}}\right)_{j-1/2}^{n-1/2} \left(t - t^{n-1/2}\right)^a dt \\
&= -\sum_{a=0}^{N_M-Z} \frac{1}{a!} \left(f_{i,x^Z t^{Z+a}}\right)_{j-1/2}^{n-1/2} \int_{t_n-1/2}^{t_n} \left(t - t^{n-1/2}\right)^a dt \\
&= -\frac{\Delta t}{2} \sum_{a=0}^{N_M-Z} \frac{2^a}{(a+1)!} \left(f_{i,\bar{x}^a \bar{t}^{Z+a}}\right)_{j-1/2}^{n-1/2} \left(\frac{4}{\Delta x}\right)^Z .
\end{aligned}
$$

The same procedure is applied to the other line segments: CD, DE, EF, and FA. It should be noted that it is not necessary to integrate $(u_i)_j^n$ over the line segment DA because it is an internal surface and is canceled out when $(u_i^*)_j^n$ over the line segment AD. The result of the remaining line integrations are:

$$\int_{CD} \left(-u^*_{i,x^Z}, f^*_{i,x^Z}\right) \cdot (dx, dt) = -\frac{\Delta x}{2} \sum_{a=0}^{N_M - Z} \frac{2^a}{(a+1)!} \left(u_{i,\bar{x}^{a+Z}}\right)_{j-1/2}^{n-1/2} \left(\frac{4}{\Delta x}\right)^Z,$$

$$\int_{DE} \left(-u^*_{i,x^Z}, f^*_{i,x^Z}\right) \cdot (dx, dt) = -\frac{\Delta x}{2} \sum_{a=0}^{N_M - Z} \frac{(-1)^a 2^a}{(a+1)!} \left(u_{i,\bar{x}^{a+Z}}\right)_{j+1/2}^{n-1/2} \left(\frac{4}{\Delta x}\right)^Z,$$

$$\int_{EF} \left(-u^*_{i,x^Z}, f^*_{i,x^Z}\right) \cdot (dx, dt) = \frac{\Delta t}{2} \sum_{a=0}^{N_M - Z} \frac{2^a}{(a+1)!} \left(f_{i,\bar{x}^a t^{Z+a}}\right)_{j+1/2}^{n-1/2} \left(\frac{4}{\Delta x}\right)^Z,$$

$$\int_{FA} \left(-u^*_{i,x^Z}, f^*_{i,x^Z}\right) \cdot (dx, dt) = \frac{\Delta x}{2} \sum_{a=0}^{N_M - Z} \frac{2^a}{(a+1)!} \left(u_{i,\bar{x}^{a+Z}}\right)_{j}^{n} \left(\frac{4}{\Delta x}\right)^Z$$

By summing over all line segments the value of u_{i,x^Z} is calculated at the new time step and is equal to:

$$\begin{aligned}(u_{m\bar{x}^Z})_j^n = {} & \frac{1}{\Delta x}\iint s_{m\bar{x}^Z} dV + \frac{1}{2}\sum_{k=0}^{N_M-Z}\frac{2^k}{(k+1)!} \\ & \left(\left[u_{m\bar{x}^{k+Z}} + \frac{\Delta t}{\Delta x} f_{m\bar{x}^Z \bar{t}^k}\right]_{j-1/2}^{n-1/2} + \left[(-1)^k u_{m\bar{x}^{k+Z}} - \frac{\Delta t}{\Delta x} f_{m\bar{x}^Z \bar{t}^k}\right]_{j+1/2}^{n-1/2}\right) - \\ & \sum_{k=1}^{\frac{N_M-Z-1}{2}} \frac{2^{2k}}{(2k+1)!}\left(u_{m\bar{x}^{2k+Z}}\right)_j^n\end{aligned} \tag{2.23}$$

Equation (2.23) provides an explicit formulation for all even spatial derivatives. As long as the higher even derivative are calculated first the last term on the RHS will have already been calculated. For example, in a fourth order accurate scheme the conserved variables are $u_i, u_{i,x}, u_{i,x^2}$, and u_{i,x^3}. In this case $\left(u_{i,x^2}\right)_j^n$ will be calculated first followed by $(u_i)_j^n$. It should also be noted that the $*$ is absent from the source term. This is because the source term treatment varies when dealing with different flow physics and may not require a Taylor series expansion. Another interesting feature in Eq. (2.23) is that when the integration over surfaces AB and FA are summed together all odd derivatives cancel out. This in turn allows for all even derivatives to be calculated first followed by all odd derivatives.

2.4 Odd Derivatives

In order to compute the odd derivatives a central differencing approach is applied following the c-τ scheme. There are two possible formulations for the odd derivatives (i) the standard c-τ scheme which is applicable if there are no discontinuities present and (ii) a weighted c-τ scheme which is used if there are discontinuities in the flow field.

To mitigate the dissipation as the local CFL number decreases the central differencing is applied at points P^+ and P^-, where $P^\pm$ are points located at

$$x_j\left(P^+\right) = x_j + (1+\tau)\frac{\Delta x}{4} = x_{j+1/2} - (1-\tau)\frac{\Delta x}{4}, \tag{2.24}$$

$$x_j\left(P^-\right) = x_j - (1+\tau)\frac{\Delta x}{4} = x_{j-1/2} + (1-\tau)\frac{\Delta x}{4}. \tag{2.25}$$

Where τ is a function of the local CFL number.

First let $(u^*_{m\bar{x}^I})^n_j(P^\pm)$ be the Taylor series expansion of $(u_{m\bar{x}^I})^j_n$ from (x_j, t^n) to $x(P^\pm)$. Then $(u_{m\bar{x}^{I+1}})^n_j$ may be solved for by subtracting $(u^*_{m\bar{x}^I})^n_j(P^-)$ from $(u^*_{m\bar{x}^I})^n_j(P^+)$ resulting in

$$(u_{m\bar{x}^{I+1}})^n_j = \frac{(u^*_{m\bar{x}^I})^n_j(P^+) - (u^*_{m\bar{x}^I})^n_j(P^-)}{2(1+\tau)} - \sum_{k=1}^{\frac{N_M-1-I}{2}} \frac{1}{(2k+1)!}(u_{m\bar{x}^{2k+1+I}})^n_j(1+\tau)^{2k}, \tag{2.26}$$

for $I = 0, 2, 4, \ldots, N_M - 1$ and $m = 1, 2, \ldots, N_{eq}$.

Since not all of the Taylor series coefficients used in calculating $(u^*_{m\bar{x}^I})(P^\pm)$ are known it is approximated by $u'_{m\bar{x}^I}(P^\pm)$. Where $u'_{m\bar{x}^I}(P^\pm)$ is the Taylor series expanded from $(x_{j\pm1/2,t^{n-1/2}})$ to $x(P^\pm)$. In this Taylor series expansion the value of u_i at points $P^\pm$ are calculated from values at the previous time step.

$$(u_{m\bar{x}^{I+1}})^n_j = \frac{u'_{m\bar{x}^I}(P^+) - u'_{m\bar{x}^I}(P^-)}{2(1+\tau)} - \sum_{k=1}^{\frac{N_M-1-I}{2}} \frac{1}{(2k+1)!}(u_{m\bar{x}^{2k+1+I}})^n_j(1+\tau)^{2k},$$

When discontinuities are present in the flow field a re-weighting and or limiting of the derivatives is required. The first step in computing the odd derivatives is to apply a weighting algorithm and then check for smoothness. If the results are not found to be sufficiently smooth a limiter, such as minmod, is applied which further suppresses dispersion.

In[3] Chang showed that the previous derived weighting schemes originally presented in[61] are applicable without modification to the new high-order CESE scheme.

For brevity only the W1 scheme from[61] is presented. First the function W is given as

$$W_{\pm}(x_-, x_+, \alpha) = \frac{|x_{\mp}|^{\alpha}}{|x_-|^{\alpha} + |x_+|^{\alpha}}. \tag{2.27}$$

To remain stable in the presence of discontinuities α is an adjustable parameter ≥ 1. The odd derivatives are now defined by:

$$\left(u_{m\bar{x}^I}\right)_j^n \equiv \left(\omega_{m-}\right)_I \left(\hat{u}_{m\bar{x}-^I}\right) + \left(\omega_{m+}\right)_I \left(\hat{u}_{m\bar{x}+^I}\right), \tag{2.28}$$

where

$$\left(\omega_{m\pm}\right)_I = W_{\pm}(u^c_{m\bar{x}-^I}, u^c_{m\bar{x}+^I}, \alpha), \tag{2.29}$$

with

$$\hat{u}_{m\bar{x}\mp^I} \equiv \pm \frac{\left(u_{m\bar{x}^{I-1}}\right)_j^n - u'_{i,x^{I-1}}(P^{\mp})}{1+\tau},$$

$$u^c_{m\bar{x}^I\mp} \equiv \pm\frac{1}{2}((u_{i,x^{I-1}})_j^n - (u'_{i,x^{I-1}})^n_{j\mp 1/2}),$$

where $(u'_{i,x^{I-1}})^n_{j\pm 1/2}$ is the Taylor series expansion from $(x_{j\pm-1/2}, n-1/2)$ to $(x_{j\pm-1/2}, n)$.

For solvers that are greater than fourth-order it was found that the weighting scheme was not always enough for stability. In these cases a smoothness criteria was established and the minmod limiter was applied as needed. The smoothness criteria used was

$$\left|\frac{\partial^{k+1}u_i}{\partial\bar{x}^{k+1}}\right| > \left|\frac{4\beta}{k+1}\frac{\partial^k u_i}{\partial\bar{x}^k}\right| \text{ for } k > 1 \tag{2.30}$$

where β is an adjustable parameter on the order of 0.1 or 0.01. If Eq. (2.30) is found to be true then the minmod limiter was applied. This smoothness check was applied to all derivatives, even and odd, greater than 1. This test ensures that each successive term in the Taylor series is a correction on the previous term. When a strong discontinuity is present Taylor series will diverge and each successive term in the series will increases. The minmod operator is defined as:

$$\text{minmod}(a, b) = \max(\min(1, a/b), 0). \tag{2.31}$$

So $u_{\bar{x}^k}$ is equal to

$$u_{\bar{x}^k} = \hat{u}_{m\bar{x}+^k}\text{minmod}(\hat{u}_{m\bar{x}-^k}, \hat{u}_{m\bar{x}+^k}) \tag{2.32}$$

The above equations provide an explicit formulation for the odd spatial derivatives when discontinuities are present in the flow field.

2.5 Comparison to the Godunov Scheme

To better understand the CESE scheme it is useful to compare it to a standard upwind scheme. The Godunov scheme[**?**] was selected because it was the first Finite Volume conserved scheme for nonlinear conservation laws and laid the foundation for current and future Finite Volume schemes. To begin, the control volume used to determine $u(i, n+1)$, where i is the spatial location and n is the time step, is shown in Fig. (2.5). In this figure the open circles are the location where the solution is stored, the solid lines are the boundaries of the control volume and the dashed lines represent the domain in which each solution is valid.

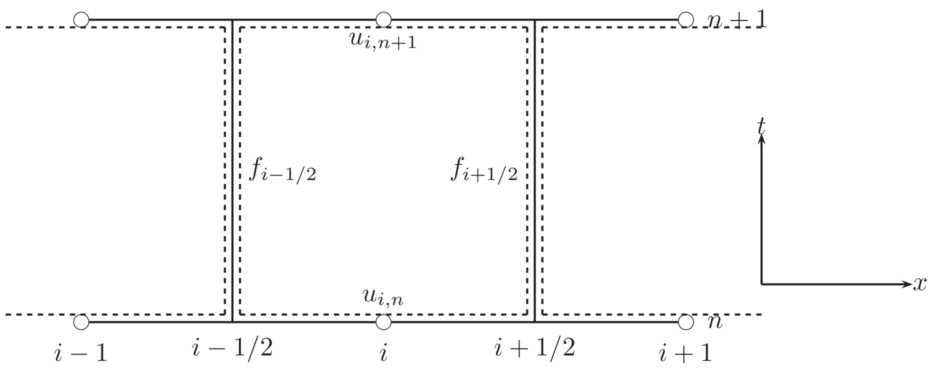


Figure 2.5: The control volume used by the Godunov scheme.

Given this control volume an expression to determine the value of u at $n+1$ would be expressed as:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t^{n+1})dx = \int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t^{n})dx + \int_{t_n}^{x_{t+1}} f(x_{i-1/2},t)dt - \int_{t_n}^{x_{t+1}} f(x_{i+1/2},t)dt, \tag{2.33}$$

where f is the flux and u is the conserved variable. It is assumed that the flux is a function of u. This equation is commonly written as:

$$\bar{u}_i^{n+1} = \bar{u}_i^n + \frac{\Delta t}{\Delta x}\left(F_{i-1/2} - F_{i+1/2}\right), \tag{2.34}$$

where $\bar{u}_i$ is the average value of u between $x_{i-1/2}$ and $x_{i+1/2}$. And $F_{i\pm1/2}$ is the flux function computed by solving the Riemann problem at the cell interface, given the values of $\bar{u}$ on the left and right side of the face.

The difference between the Godunov and CESE schemes are apparent when one compares their respective control volume and resulting integral equations. The most obvious differences are that the CESE scheme uses a staggered mesh and requires two separate time integration to progress one full time step. Another major difference is that the Godunov scheme requires the solution of a Riemann problem when integrating along the boundaries $[(x_{i\pm1/2},t^n),\ (x_{i\pm1/2},t^{n+1})]$. This is due to having two different values of u, i.e. u_L and u_R, on either side of the boundary of the control volume. Whereas the CESE scheme has only one valid expression of u along each boundary of its control volume or CE.

2.6 Numerical Outline

To summaries the numerical procedure used is as follows:

1. Calculate all of the Taylor series coefficients for the flux and the temporal derivatives of the conserved variables at $(n-1/2, j \pm 1/2)$, points C and E in Fig. (2.4).

2. Calculate all even derivatives using Eq. (2.23), starting with the highest unknown even derivative.

3. Calculate the highest odd derivative via Eq. (2.28)

4. Check for smoothness using Eq. (2.30) and apply minmod, Eq. (2.32), as needed

5. Repeat steps 3 and 4 until all odd derivatives are found

6. Check the even derivative for smoothness, Eq. (2.30), and apply the minmod, Eq. (2.32) limiter as needed. This is done for all even derivatives except the zero derivative, i.e. k=0 in Eq. (2.32).

2.7 One Dimensional Results

The following test cases show how the CESE method improves as the order of accuracy increases. The test cases used to verify the order of convergence are: (i) the linear convection equation with a source term, (ii) an acoustic wave modeled by the linearized isentropic Euler equations, and, (iii) an acoustic wave modeled by the non-linear Euler equations. For all three cases, the order of accuracy was calculate through the following formula:

$$L_2 \equiv \sqrt{\int |\phi|^2 dx} \approx \sqrt{\sum_i |\phi_i|^2 \Delta x_i}$$

where ϕ_i is defined as the difference between the analytical and numerical solution and Δx_i is the grid spacing at a given location i. In all cases Δx is constant. The rate of convergence is taken as the slope of the best fit line through the points ($log_{10}(\Delta x)$, $log_{10}(L_2)$).

2.7.1 Convection Equation with Source Term

The first test case is the linear convection equation with a source term. In this problem the following equation is solved

$$
\begin{aligned}
&\frac{du}{dt} + a\frac{du}{dx} = aS_0\cos(x) \\
&\quad -2\pi < x < 2\pi; t > 0
\end{aligned}
$$

where a and S_0 are constant. Under the periodic boundary condition an analytical solution to this problem is

$$
u(x,t) = cos(x - at) + S_0 sin(x).
$$

This governing equation contains a source terms which needs to be integrated. Since the source term is linear an analytical solution is found that is only dependent on x.

$$
\begin{aligned}
&S = aS_0\cos(x) \\
&\iint S\,dx\,dt = aS_0\frac{\Delta t}{2}\sin(x)|_{x_{j-1/2}}^{x_{j+1/2}}, \\
&\iint S_{\bar{x}\bar{x}}\,dx\,dt = -aS_0\left(\frac{\Delta x}{4}\right)^2\frac{\Delta t}{2}\sin(x)|_{x_{j-1/2}}^{x_{j+1/2}} \quad ,\ldots, \\
&\iint S_{\bar{x}^{2n}}\,dx\,dt = (-1)^{n/2}aS_0\left(\frac{\Delta x}{4}\right)^{2n}\frac{\Delta t}{2}sin(x)|_{x_{j-1/2}}^{x_{j+1/2}},\ \text{for } n = 0,1,\ldots,\frac{N_M - 1}{2}.
\end{aligned}
$$

For the convergence tests, $a = S_0 = 1$ and the test time was set to $2.5\frac{l}{a}$, where l is the length of the domain. In all calculations, CFL $= 0.7$. For these simulations the

Table 2.1: The convergence rates for the numerical solutions of the convection equation, and the averaged, normalized time for case with varying CESE solver orders.

Order of CESE Solver	Numerical Order	Normalized Time
2	2.00	1.00
4	4.01	5.22
8	7.95	23.65
12	12.12	59.04
16	16.05	115.83
20	20.09	190.95

order increases from second to twentieth. The upper limit for the order of the solver was set by the accuracy of a double precision variable. In other words the difference between the analytical and numerical solution was smaller than what can be resolved by a 64bit variable. Shown in Fig. (2.6) and Table (2.1), the actual convergence rate agrees well with the order of accuracy of the scheme employed.

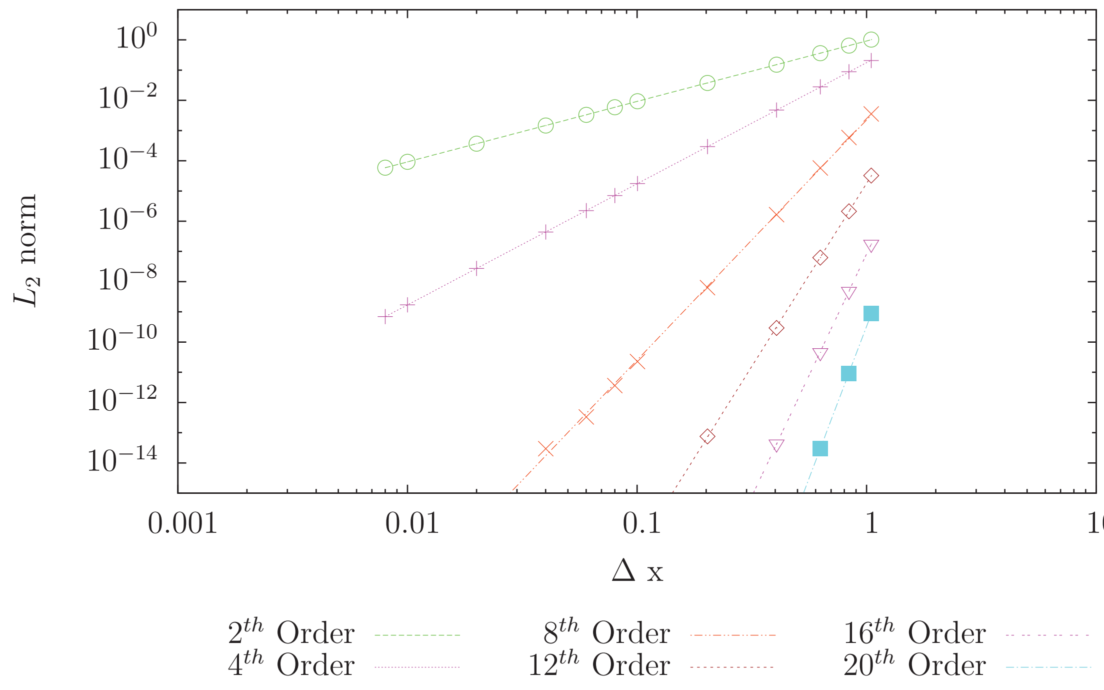


Figure 2.6: The L_2 norm of numerical solutions of the convection equation with source term. The symbols represent the actual calculated data and the lines represent the best-fit curves of the data.

Another important aspect to consider is whether the results from a high-order simulation is worth the additional computational cost. To determine this the L_2 norm is plotted against the CPU time required to complete the simulation. These results are shown in Fig. (2.7). From this plot it is easily shown that when solving a linear convection equation with a source term that it is more numerically efficient to increase the order of accuracy than increase the resolution.

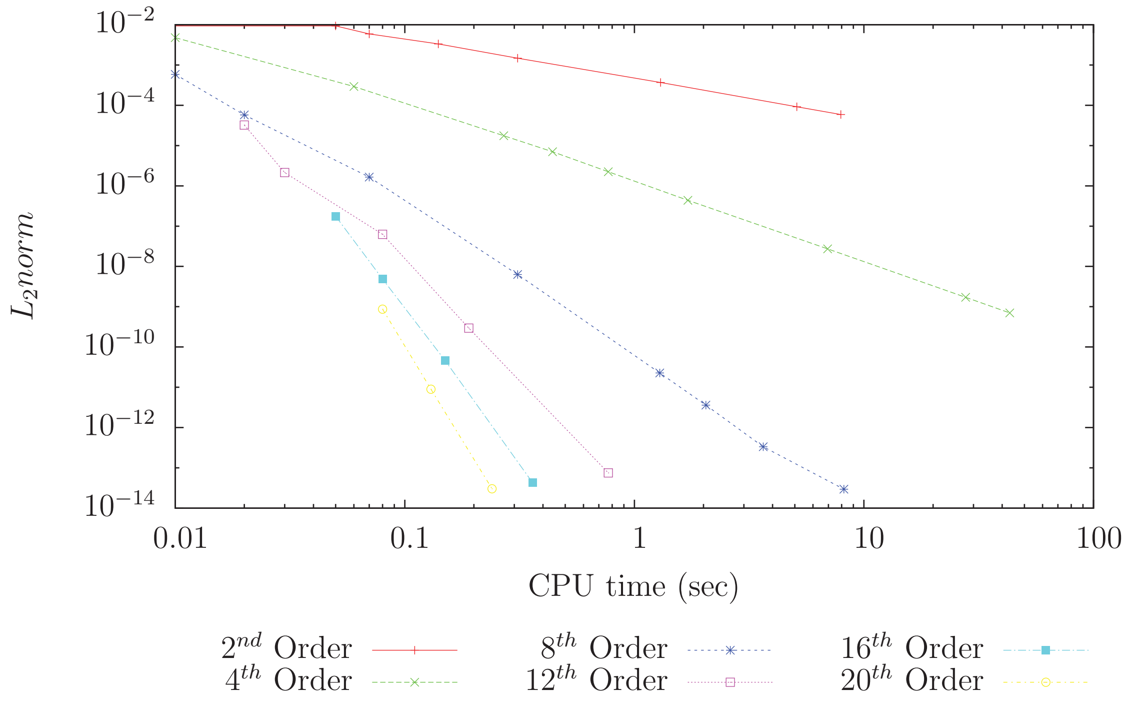


Figure 2.7: The L_2 norm versus the computational time for the solution of the convection equation.

2.7.2 Linear Acoustic Equation

The second test case uses the linearized isentropic Euler equations to solve an acoustic wave. The governing equation is expressed as:

$$\mathbf{U} = (u_1, u_2)^T = (\rho, v)^T ,$$

$$\mathbf{F} = \left(\rho_\infty v, \frac{a_\infty^2}{\rho_\infty}\rho\right)^T = \left(\rho_\infty u_2, \frac{a_\infty^2}{\rho_\infty}u_1\right)^T$$

where ρ, U, and a are respectively the density, velocity, and speed of sound. The speed of sound is equal to $\sqrt{\gamma p/\rho}$ with γ=1.4. The values with a subscript ∞ are mean values of the flow variables. The Jacobian matrix for the governing equation is:

$$\begin{bmatrix} 0 & \rho_\infty \\ \frac{a_\infty^2}{\rho_\infty} & 0 \end{bmatrix}$$

Since all of the first derivatives are constant the higher derivatives are zero. This reduces the calculation of all fluxes to a matrix vector multiplication.

Under periodic boundary conditions an analytical solution for the linearized acoustic wave equation is

$$\begin{aligned}
\rho &= \rho_\infty + \frac{\varepsilon \rho_\infty}{a_\infty} cos\left(\frac{2n\pi}{l}(x - a_\infty t)\right) \\
U &= U_\infty + \varepsilon cos\left(\frac{2n\pi}{l}(x - a_\infty t)\right) \\
&\text{for } \frac{-l}{2} < x < \frac{l}{2}; t > 0
\end{aligned}$$

where n, l, and ε are respectively the number of waves in the domain, length of the domain, and an amplification factor. For this test case p_∞=1, ρ_∞=γ, ε=10^{-2}, n=1, and l=2. The run time was equal to $4.25\frac{l}{a_\infty}$ which allows the wave to propagate through the domain 4.25 times. The CFL number is constant throughout the domain and is equal to 0.75. As seen in Fig. (2.8) the desired order of convergence closely matches the actual order of convergence. Table (2.2) shows the desired order of convergence, the actual order of convergence, and the normalized time. Figure (2.9) shows that it is typically more efficient to increase the order of accuracy of the solver than resolving a mesh. When the L_2 norm is still relatively high, $OD(10^{-8})$ it was found that eighth through twentieth order schemes had approximately the same computational efficiency. The relative numerical cost was calculated by taking the average simulation time per cell per iteration for multiple resolutions and dividing it by the cost of the second-order version.

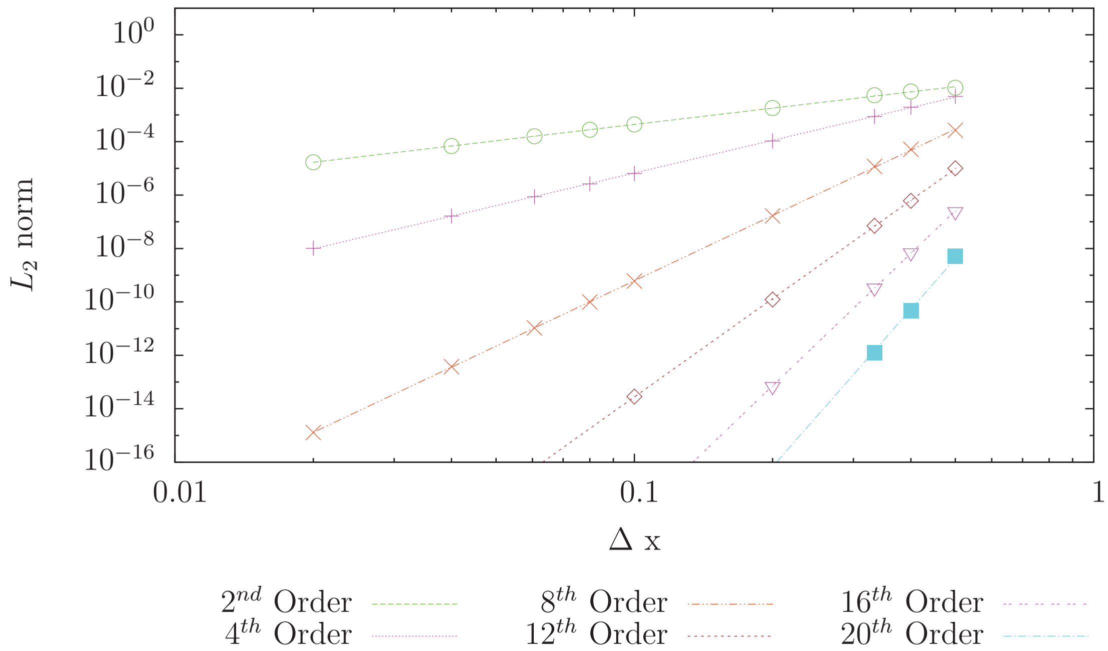


Figure 2.8: The L_2 norm for the Linear acoustic simulation, the points are actual calculated data and the line is a best-fit cure of the data.

Table 2.2: Convergence rates for the linear acoustic solver and the average normalized time for case.

Order of CESE Solver	Numerical Order	Normalized Time
2	2.03	1.00
4	4.06	3.43
8	8.12	14.47
12	12.21	34.33
16	16.48	67.98
20	20.52	116.89

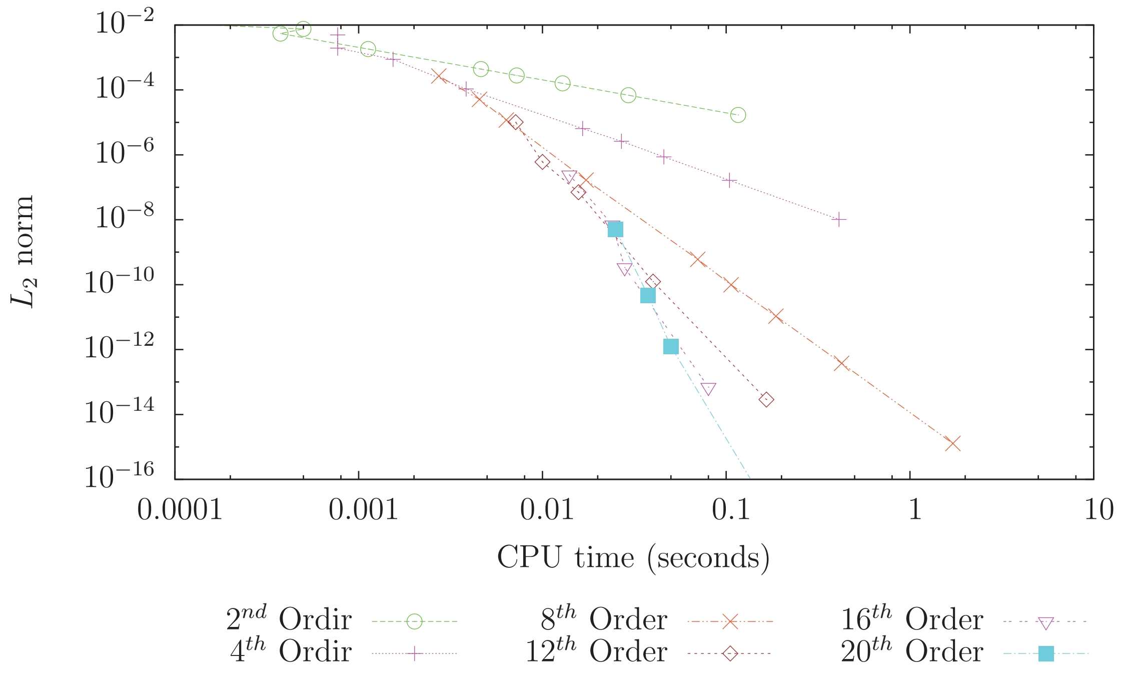


Figure 2.9: The L_2 norm versus the computational time for the solutions of the linear acoustic wave equations.

The above cases showed that the new scheme achieved higher-order convergence for a coupled linear wave equations.

2.7.3 Euler Equations

In this section the higher order CESE scheme is tested for its ability to achieve higher order convergence when solving the non-linear coupled Euler equations with a smooth solution. The schemes ability to accurately resolve discontinuities in a flow field will also be examined.

To show higher order convergence for a non-linear equation the acoustic equations are solved. There are some difficulties when using the analytical solution because the Euler solver will capture the non-linearities present in the flow field while the linear solution does not. This will lead to increasing errors in the analytical solution as Δx decreases. To mitigate this error, the perturbation was reduced to 10^{-6}. For this test case p_∞=$1/\gamma$, ρ_∞=1, n=2, and l=4 and the simulation time is $2.5\frac{l}{a_\infty}$. The CFL number is almost constant throughout the domain and is equal to 0.8. The convergence rates are shown in Table 2.3. Figure (2.10) shows the convergence rates for the second-, fourth-, eighth- and twelfth-order schemes. From this figure it is obvious that for a given mesh a higher order scheme has a lower L_2 norm then a scheme with a lower order of convergence. Although if the simulation time is taken into consideration the advantage of a high-order scheme is not as clear. These results are shown in Fig. (2.11), in this figure the L_2 norm is compared to the CPU time required to complete the simulation. With the exception of the first run all high-order schemes out performed the second-order scheme. Overall the fourth- and eighth-order scheme tends to be the most efficient schemes. This is in contrast to the linear schemes where it was always more efficient to use a high-order scheme then to use a more refined mesh. This was most likely due to the increased computation cost associated with flux calculation.

Table 2.3: Convergence rates of the numerical solutions of the convection equation, and the averaged, normalized time for case with different order of accuracy.

Order of CESE Solver	Numerical Order	Normalized Time
2	2.00	1.00
4	3.97	2.84
8	7.48	12.40
12	11.96	32.30

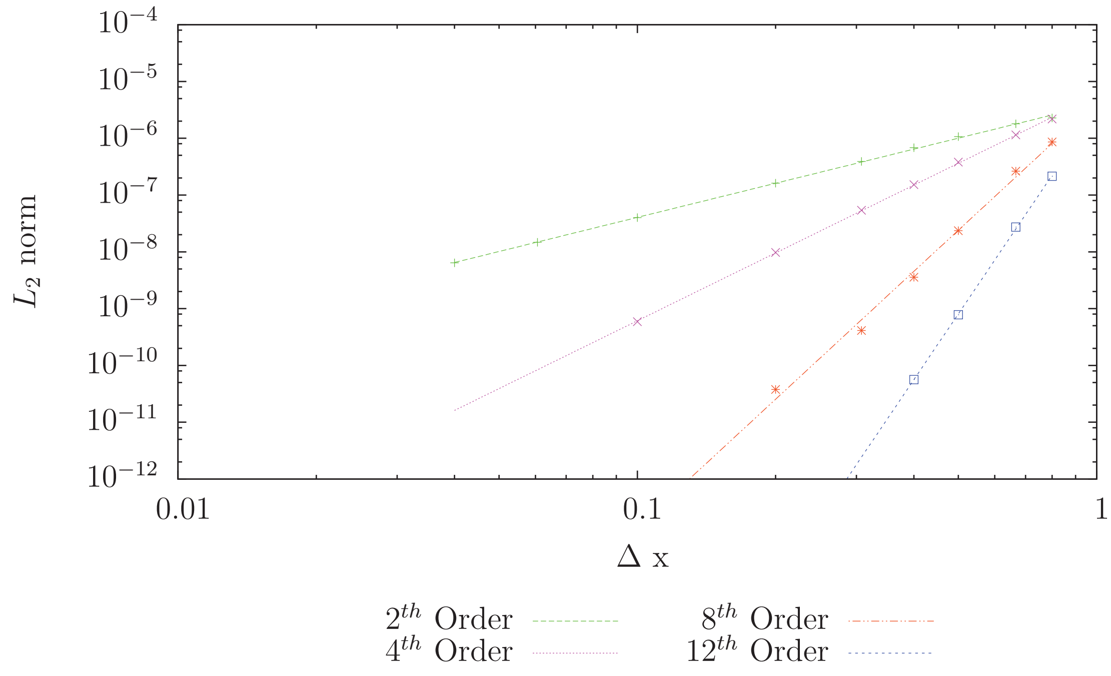


Figure 2.10: The L_2 norm of the numerical solutions of the Euler solver for solving the acoustic waves.

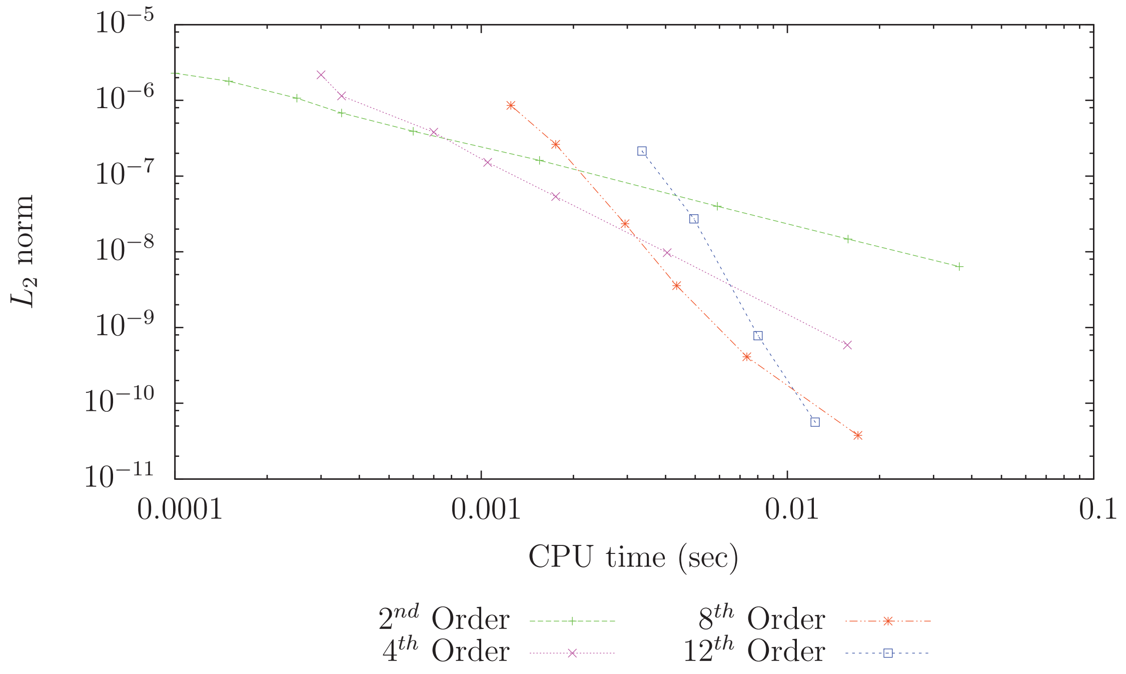


Figure 2.11: The L_2 norm versus the computational time for the solution to the non-linear Euler solver.

Next the high-order CESE methods ability to accurately resolve shocks and contact discontinuities is demonstrated. To accomplish this two standard test cases where run. A parametric study was performed by varying both the spatial resolution and the order of accuracy of the solver. Since neither of these simulations has an analytical solution to compare against, a converged solution was used as a reference. To obtain a converged solution the second-order CESE scheme was used on a very fine mesh. To provide another source of comparison the fifth-order space third-order time monotonicity preserving (MP53) method [69] was also used.

The first test case is the Woodward's blast wave problem[70]. Woodward's blast wave problem consists of two shock waves of different strengths heading towards each other with wall boundary conditions. The simulation ran for a non dimensional time of 0.47 and the initial conditions are

$$(u,\rho,p) = \begin{cases} (0, \quad 1, \quad 10^{3}) & 0 < x < 0.1 \\ (0, \quad 1, \quad 10^{-2}) & 0.1 < x < 0.9 \\ (0, \quad 1, \quad 10^{2}) & 0.9 < x < 1.0 \end{cases}$$

The second test case is the Shu Osher's problem[71]. In this simulation a Mach 3 shock moves to the right and collides with a sinusoidal entropy disturbance moving to the left. The boundary conditions are non-reflective and the initial conditions are

$$(u,\rho,p) = \begin{cases} (2.629369, \quad 3.857143, \quad 10.3333) & -1 < x < -0.8 \\ (0, \quad 1 + 0.2\sin(5\pi x), \quad 1) & -0.8 < x < 1.0 \end{cases}$$

$$0.0 < t < 0.47.$$

For both simulations α in Eq. (2.27) varies by this equation.

$$\alpha = \min\left(\frac{1}{2}\left|\left(\frac{u^{c}_{m\bar{x}^{z}+}}{u^{c}_{m\bar{x}^{z}-}} + \frac{u^{c}_{m\bar{x}^{z}-}}{u^{c}_{m\bar{x}^{z}+}}\right) - 1\right|, 1.0\right).$$

Using this equation the value of α approaches zero as $u^c_{m\bar{x}^z+}$ approaches $u^c_{m\bar{x}^z-}$. β in Eq. (2.30) was set to 0.01.

Figure (2.13) shows the results from the Shu-Osher simulation. In this figure the CESE scheme was run at second-, fourth-, sixth-, and eighth-order and at a resolution of 400 and 800 uniform cells. For comparison each density trace compares one of the CESE simulation to the MP53 scheme. The black line denotes a converged solution obtained by running the second-order CESE scheme with 3201 uniform cells. For the lower resolution case the MP scheme outperforms the second-order CESE scheme and compares favorably to the fourth- and sixth-order scheme. The eighth-order CESE scheme also compares favorably to the MP53 scheme except at the shock where the CESE scheme has some dispersion. For the case with 800 cells the resolution is high enough that the difference between the different various simulations is not apparent. Again, the exception to this is that the eighth-order CESE simulation has more dispersion around the shock than the other schemes.

The results for the Blastwave simulation are shown in Fig. (2.13). For this simulation the CESE scheme was run at an second-, fourth-, sixth-, and eighth-order. These simulations where run at a resolution of 400 and 800 uniform cells. The black line represents a converged solution obtained by the running the second-order CESE scheme with 20001 equally spaces cells. For this figure each plot compares the CESE results to its next higher order of accuracy. For example the top plot compares the second-order to the fourth-order and the next one down compares the fourth-order to the sixth-order and so on. For all simulations the shock was resolved with one or two points with no noticeable dispersion. For the 400 cell case the resolution of the left side contact discontinuity saw no noticeable improvement whereas the right side contact, located between $0.75 < x < 0.8$, features saw a continuous improvement from second- to fourth- to sixth-order. On the other hand the eighth-order CESE simulation has more dissipation around the right side contacts than the second-order simulation. The simulations with 800 cells have some interesting results as well. For example the results for the fourth-order CESE scheme are by far worse than the results from the second-order simulation. But the sixth- and eight-order simulations slightly out perform the second-order simulation. The poor results from the fourth-order simulation could be caused by switching to the minmod limiter too early which could be rectified by using a more aggressive weighting scheme.

In summary the results from the convergence test cases showed that the CESE scheme is able to achieve the desired order of accuracy for both linear and non-linear simulations. Furthermore, it was shown that it was always more computationally efficient to increase the order of accuracy than to add more cells for linear systems. For the Euler equations the fourth and higher order solvers where more efficient than the second-order CESE solver. When comparing the fourth and higher order solvers the efficiency of the solvers was not apparent until higher resolutions. The results from the Blastwave and Shu-Osher simulations showed that the high-order CESE solvers are able to accurately resolve discontinuities while providing accurate results in smooth regions. Furthermore, this was accomplished without using any physics specific treatment. This makes these schemes more portable to new physical systems.

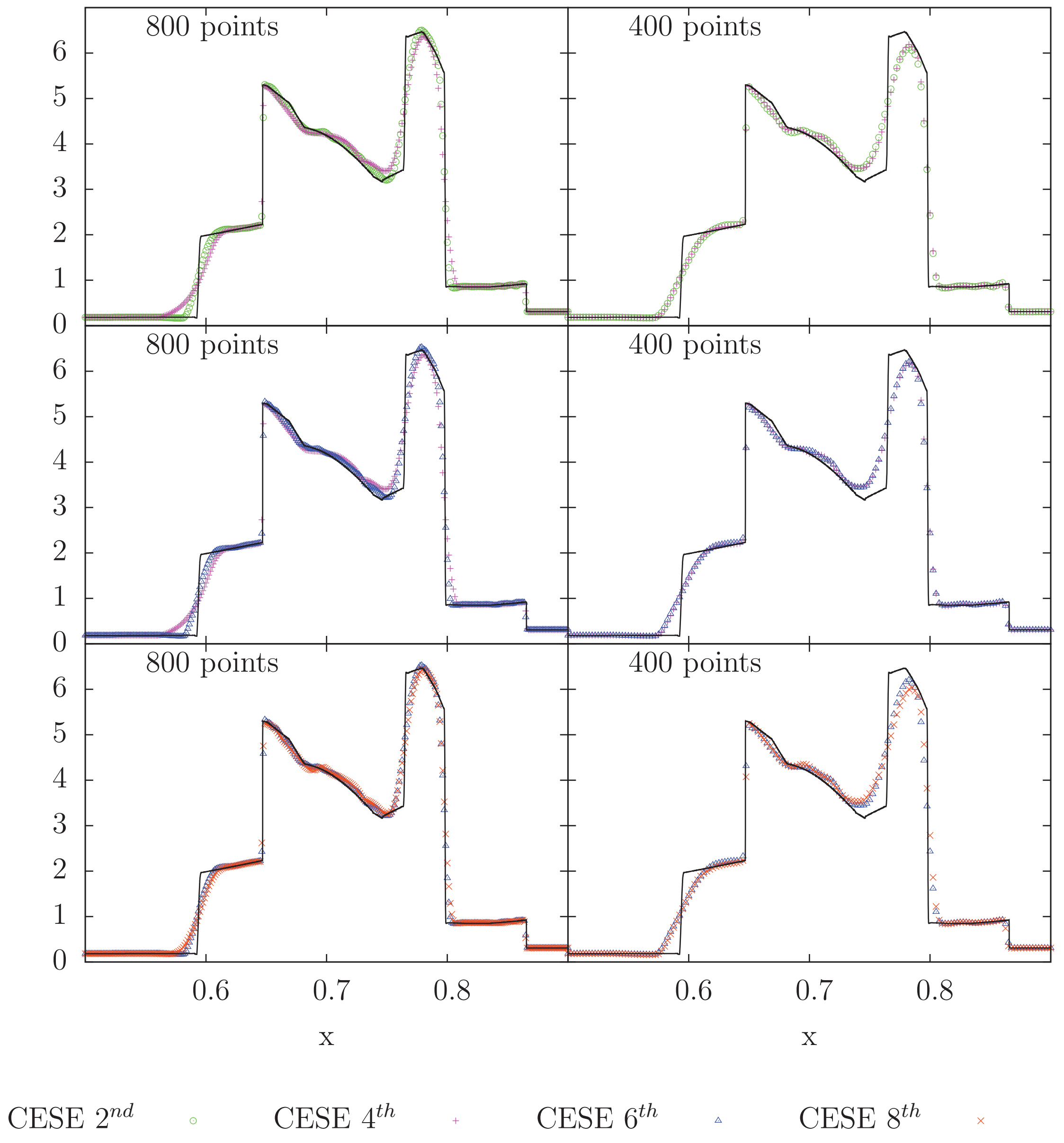
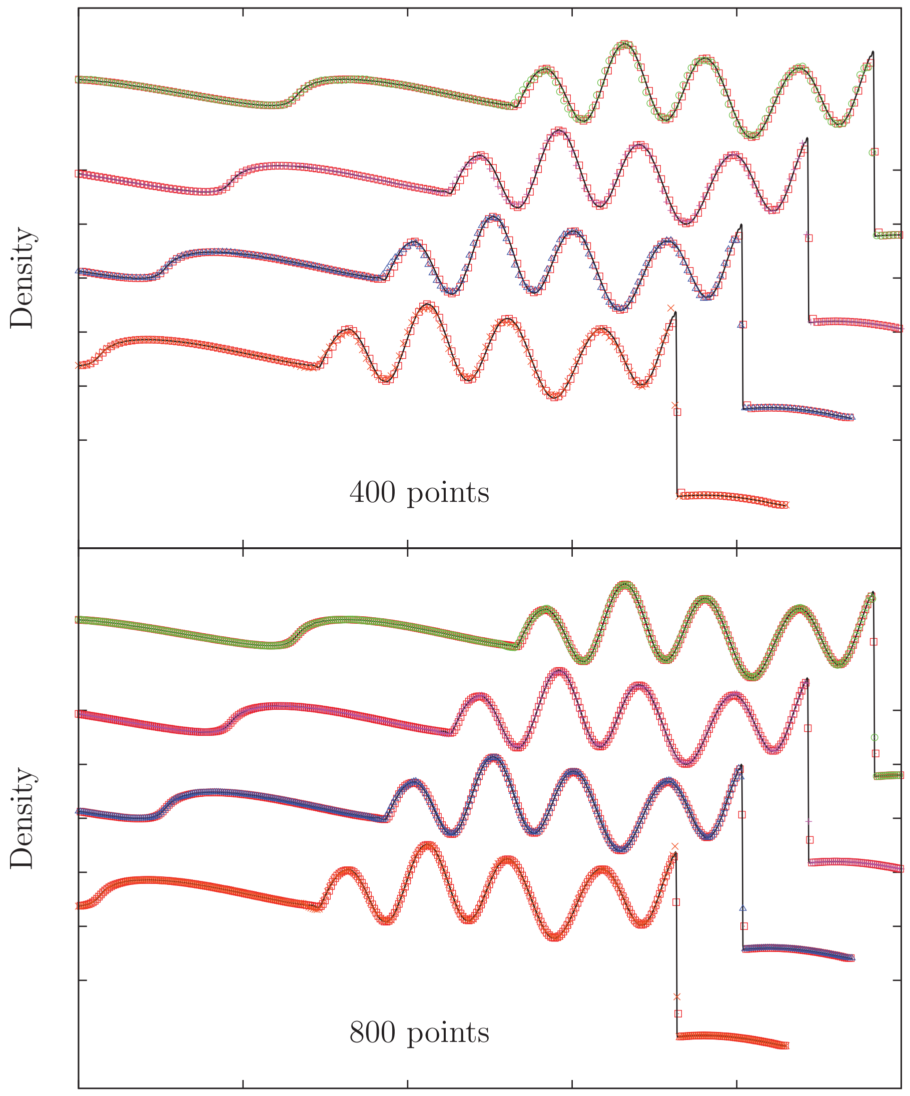


Figure 2.12: Plots of the density profiles of the Woodwards blast wave problem. The converged simulation was done by using the second-order CESE with 20001 points. For better presentation, only a subset of the domain is shown in these plots.



MP53 □ CESE 4th + CESE 8th ×
CESE 2nd ○ CESE 6th △

Figure 2.13: Plots of the calculated density profiles for Shu and Osher's problem. Each plot has a different spatial resolution. The converged simulation was done by using the second-order CESE scheme with 3201 points.

CHAPTER 3

TWO-DIMENSIONAL CESE

This chapter details the derivation of the two-dimensional high-order CESE scheme. The equations derived in this chapter are valid for any order of accuracy and for both triangular and quadrilateral meshes. To begin, the two-dimensional hyperbolic equations are cast into the following vector form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^x}{\partial x} + \frac{\partial \mathbf{F}^y}{\partial y} = 0 \tag{3.1}$$

where $\mathbf{U}$ is the unknown vector and $\mathbf{F}^{x,y}$ is the flux vector in the x and y direction, respectively:

$$\mathbf{U} = (u_1, u_2, \ldots, u_i)^T,$$

$$\mathbf{F}^{x,y} = (f_1^{x,y}, f_2^{x,y}, \ldots, f_i^{x,y})^T.$$

The superscript T denotes transverse and m is the number of governing equations in the system. The hyperbolic equations are cast into a divergence-free format:

$$\nabla \cdot \mathbf{h}_i = 0, \tag{3.2}$$

where $\mathbf{h}_i = (f_i^x, f_i^y, u_i)^t$ is the space-time flux vector and $i = 1, \ldots, m$. Here, the divergence operates in the three-dimensional Euclidean space $\mathcal{E}_3(x, y, t)$. Aided by Gauss' theorem, Eq. (3.2) becomes a surface integration of the space-time flux vector $\mathbf{h}_i$:

$$\oint_{S(R)} \mathbf{h}_i(x, y, t) \cdot d\mathbf{S} = 0. \tag{3.3}$$

where $d\mathbf{S} = \mathbf{n} d\sigma$ is a surface element with $\mathbf{n}$ as the unit normal vector pointing outward of the three-dimensional space-time region R and $d\sigma$ as the differential area as a part of $S(R)$, i.e., the surface of R. The CESE method is designed to integrate Eq. (3.3) in the space-time domain.

The rest of this chapter is organized into the following sections: (1) Discretization of the mesh and governing equations, (2) Flux and temporal derivatives calculations, (3) Two-dimensional space-time integration, (4) Two-dimensional central differencing procedure, (5) An outline of the steps required to progress the dependent variables to the next time step for a fourth-order accurate scheme, (6) Test cases.

3.1 Two-Dimensional Discretization

The geometry of CE and SE in a high-order CESE method is identical to that of the original two-dimensional second-order CESE method[59]. For completeness a full definition of these elements is defined below. Wherever possible the notation used by Wang and Chang[59] is used during the development of the two-dimensional high-order CESE scheme.

To begin a review of the CE for a two-dimensional system will be reviewed. As a reminder the primary purpose of the CEs are to enforce flux conservation. As such the CE are required to encompass the entire space-time domain without overlaps or voids. Consider mesh point (j, n) where j is the cell id and n is the current time step number. Cell j has several neighboring cells and are denoted by j_r where $r = 1, 2, 3$ for a triangular cell and $r = 1, 2, 3, 4$ for a quadrilateral mesh. Every cell is associated with three or four Basic CEs (BCEs) which forms a Compound CE (CCE). The number of associated BCEs depends on the cell type. Three for triangular meshes and four for quadrilateral meshes. A BCE is formed by taking the cell center of cell j, the two vertices associated with a single face and the neighboring cell associated with the same face. These points forms a quadrilateral in the x, y plane, next this plane is extruded in time for a distance of $\Delta t/2$. This new shape is a box in in the three-dimensional space-time domain. It is important to note that the type of mesh only affects the number of BCEs associated with a single cell and does not impact how each BCE is formed it. As an example consider the simplified triangular mesh shown in Fig. (3.1). In this figure the solid lines denote the boundaries of each cell of the imported mesh. The vertices are denoted by an open square and have the labels: 2, 4, 6, 8, 9 and 10. The center of the triangles denoted by an open circle and labeled as points 1, 3, 5 and 7. Originally the center of the triangle was defined as the centroid but some recent studies has shown that the incenter provides better results. The solution points are denoted by a cross and are labeled $1^\times$, $3^\times$, $5^\times$, and $7^\times$. The projection of the BCE onto the x, y plane are denoted by the doted line and are made up of the points: $\{1, 2, 3, 4\}$, $\{2, 5, 6, 3\}$ and $\{3, 6, 7, 4\}$. The full three-dimensional BCE is shown in Fig. (3.2b). The solution points are of critical importance to the CESE scheme and is defined as the centroid of the CCE. To determine the solution point associated with cell j let S_r be the area of the bottom face of the BCE, e.g. points $\{1, 2, 3, 4\}$, also let $\overline{X}, \overline{Y}$ be the center of the same quadrilateral. With these definitions the centroid of the CCE is:

$$\sum_{r=1}^{Nface} S_r \overline{X}_r = x_{cen} \sum_{r=1}^{Nface} S_r \qquad \& \qquad \sum_{r=1}^{Nface} S_r \overline{Y}_r = y_{cen} \sum_{r=1}^{Nface} S_r$$

where $Nface$ is the number of faces on cell j.

The second element that needs defining is the SE. As a reminder the primary purpose of the SE is to define the region in which the Taylor series expansion of each conserved variable is valid. As such the SEs may not overlap but are not required to fill up the entire domain. A SE for a triangular cell is shown in Fig. (3.2a).In this figure the SE includes three vertical rectangular surfaces, i.e., $\square\, 3'2'2''3''$, $\square\, 3'4'4''3''$,

$\square$ $3'6'6''3''$, and one horizontal hexagonal plane defined by points 125674. Where $\square$ servers as a reminder that the points are coplanar and make up a single face on the CE. These 4 planes intersect at point 3. On these four planes, the primary unknowns u_i and the fluxes $f_i^{x,y}$, with $i = 1, \ldots, m$, are discretized by a Taylor series expansion with respect to point $3^\times$. Another important feature of the SEs is that they span an entire time step whereas the CEs only span a half-time step.

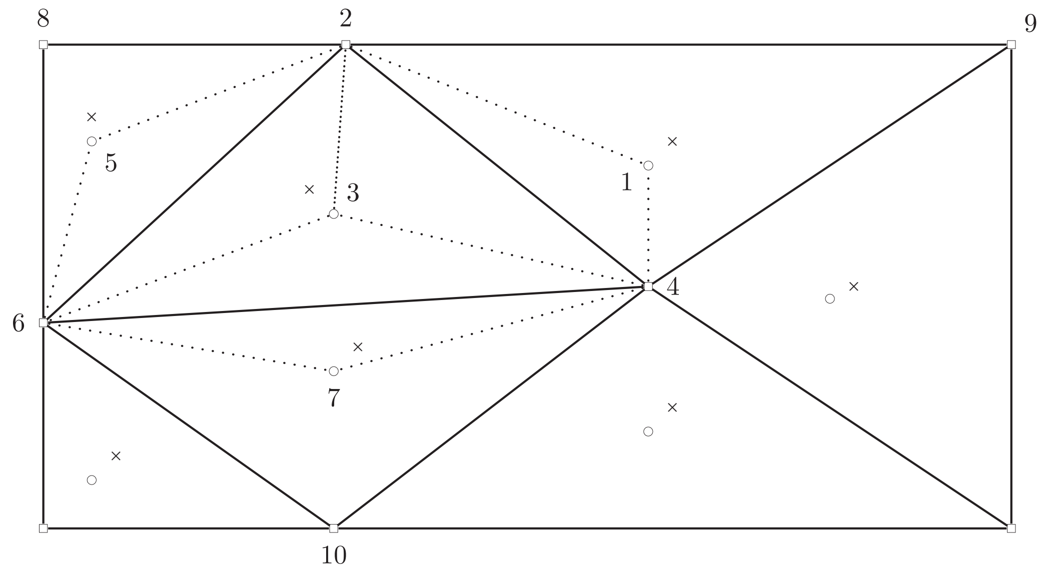
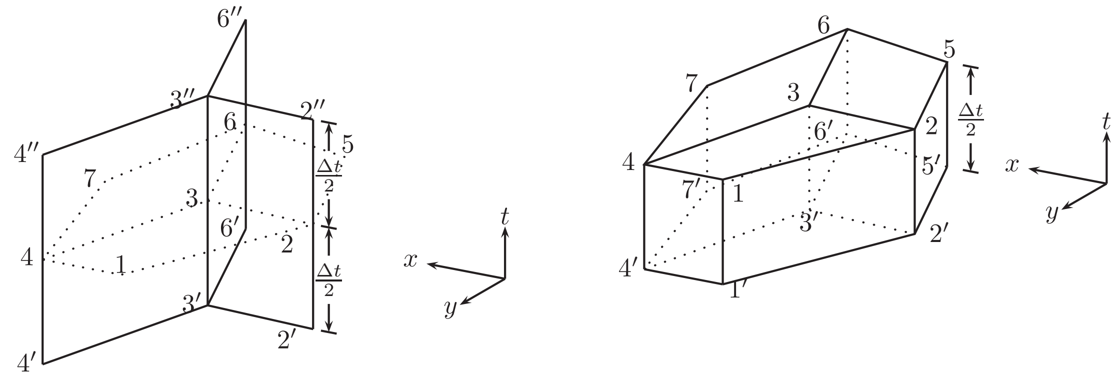


Figure 3.1: A schematic of a triangular mesh for the fourth-order CESE method. The cell vertices are marked by squares. The circles are the centroids of the triangular cells. The crosses indicate the centroid of the CCE, i.e., a hexagonal region defined by points 125674. Point $3^\times$ is also a solution point.



(a) Solution Element (b) Conservation Element

Figure 3.2: The SEs and CEs associated with point 3 (or $3^{\times}$). Mesh points in the previous time step is denoted by an apostrophe. Mesh points in the next new time step is denoted by a double apostrophe.

The next step in the discretization process is to express the conserved variables and fluxes as a Taylor series expansion anchored at the solution points. These Taylor series are expanded in both space, x, y, and time, t. For example a third-order Taylor series would be used in a fourth-order accurate CESE scheme and would be expanded as:

$$
\begin{aligned}
u_i^*(x,y,t) =& (u_i)_j^n + (u_{i,x})_j^n \Delta x + (u_{i,y})_j^n \Delta y + (u_{i,t})_j^n \Delta t + (u_{i,xt})_j^n \Delta x \Delta t + \\
& (u_{i,yt})_j^n \Delta y \Delta t + \frac{1}{2}\Big((u_{i,xx})_j^n \Delta x^2 + (u_{i,yy})_j^n \Delta y^2 + (u_{i,tt})_j^n \Delta t^2 + \\
& \Big[(u_{i,xy})_j^n + (u_{i,yx})_j^n \Big] \Delta x \Delta y \Big) + \\
& \frac{1}{6} \Big[(u_{i,xxx})_j^n \Delta x^3 + (u_{i,yyy})_j^n \Delta y^3 + (u_{i,ttt})_j^n \Delta t^3 \Big] + \\
& \frac{1}{6}\Big(\Big[(u_{i,xxy})_j^n + (u_{i,xyx})_j^n + (u_{i,yxx})_j^n \Big] \Delta x^2 \Delta y + \\
& \Big[(u_{i,xyy})_j^n + (u_{i,yxy})_j^n + (u_{i,yyx})_j^n \Big] \Delta y^2 \Delta x \Big) + \\
& \frac{1}{2} [(u_{i,xxt})_j^n \Delta x^2 \Delta t + (u_{i,xtt})_j^n \Delta x \Delta t^2 + (u_{i,yyt})_j^n \Delta y^2 \Delta t + \\
& (u_{i,ytt})_j^n \Delta y \Delta t^2] + (u_{i,xyt})_j^n \Delta x \Delta y \Delta t
\end{aligned}
\tag{3.4}
$$

The subscript j denotes the solution point j and the superscript n denotes the time step. In Fig. (3.2), point (j,n) is point $3^\times$. Thus (x_j, y_j, t^n) is the space-time coordinates of point $3^\times$. The distance to an arbitrary location (x,y,t) inside the SE from the solution point (x_j, y_j, t^n) is denoted by $\Delta x = x - x_j$, $\Delta y = y - y_j$, and $\Delta t = t - t^n$. If the flow variables u_i and their spatial and temporal derivatives are known at (x_j, y_j, t^n), then the value of u_i at any location in the SE is calculated by Eq. (3.4).

For an arbitrary space-time location (x,y,t) inside a SE, the value of $u_{i,x}(x,y,t)$

are calculated by the second-order Taylor series:

$$
\begin{aligned}
u^*_{i,x}(x,y,t) =&(u_{i,x})^n_j + (u_{i,xx})^n_j\Delta x + (u_{i,xy})^n_j\Delta y + (u_{i,xt})^n_j\Delta t + \\
&\frac{1}{2}\left[(u_{i,xxx})^n_j\Delta x^2 + (u_{i,xyy})^n_j\Delta y^2 + (u_{i,xtt})^n_j\Delta t^2\right] + \\
&\frac{1}{2}\left[(u_{i,xxy})^n_j + (u_{i,xyx})^n_j\right]\Delta x\Delta y + \left[(u_{i,xxt})^n_j\Delta x + (u_{i,xyt})^n_j\Delta y\right]\Delta t
\end{aligned}
\tag{3.5}
$$

Similarly, the values of $u_{i,y}$ and $u_{i,t}$ are expressed by a second-order Taylor series.

To proceed, for an arbitrary location (x, y, t) in a SE, the values of the second derivatives of u_i, i.e., $u_{i,xx}$, $u_{i,yy}$, $u_{i,tt}$, $u_{i,xy}$, $u_{i,yx}$, $u_{i,xt}$, and $u_{i,yt}$, are calculated by the first-order Taylor series expansion. For example,

$$
u^*_{i,xx}(x,y,t) = (u_{i,xx})^n_j + (u_{i,xxx})^n_j\Delta x + (u_{i,xxy})^n_j\Delta y + (u_{i,xxt})^n_j\Delta t \tag{3.6}
$$

All third derivatives of u_i, i.e., $u_{i,xxx}$, $u_{i,yyy}$, $u_{i,xxy}$, $\ldots$, $u_{i,ttt}$, $u_{i,ttx}, \ldots$, etc., remain constant for all locations inside a SE.

The alternate, or symmetric, rule of differentiation states that the order of differentiation can be changed, e.g. $\partial^2 u/(\partial x\partial y) = \partial^2 u/(\partial y\partial x)$. In this investigation the alternate rule of differentiation was not applied across all derivatives. For example, at point (x_j, y_j, t^n), $(u_{i,xy})^n_j \neq (u_{i,yx})^n_j$. These two terms are stored separately in the computer code. However, for derivatives involving differentiation with respect to time, the alternate rule is employed. For example, it is assume that $(u_{i,xt})^n_j = (u_{i,tx})^n_j$

and $(u_{i,xtt})^n_j = (u_{i,txt})^n_j$. This decision was spurred by many factors. The first was shown in Chang's 2010 paper, where it was shown that the symmetric property of differentiation could not be applied to odd derivatives[3]. Another factor that lead to this decision was that the spatial and temporal derivatives evaluated at the solution point are real numbers. A third reason was the way in which the values with the same number of x and y derivatives where averaged together. The final reason was that it was unknown if the even derivatives would have to be weighted in a similar fashion as in the one-dimensional CESE scheme. Due to all of the reasons it was decided to not apply the symmetric property to any of the conserved variables, e.g. u, u_x, u_{xx}, u_{yx}, u_{xy}, $\dots$.

In general, at an arbitrary location (x, y, t) inside a SE, the value of u_i is represented by the Nth-order Taylor series expansion written in the following form:

$$u_i^*(x,y,t) = \sum_{a=0}^{N}\sum_{b=0}^{N-a}\sum_{c=0}^{N-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial t^c}\right)_j^n \frac{\Delta x^a}{a!}\frac{\Delta y^b}{b!}\frac{\Delta t^c}{c!} \tag{3.7}$$

where terms with the same values of a, b, c are averaged together, e.g. $\frac{\partial^2 u_i}{\partial x \partial y} = 0.5\left(u_{i,xy} + u_{i,yx}\right)$. Moreover, at an arbitrary location (x, y, t) inside a SE, the value of of all derivatives of u_i are succinctly expressed by the following Taylor series expansion:

$$\frac{\partial^C u_i(x,y,t)}{\partial x^I \partial y^J \partial t^K} = \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^B u_i^*}{\partial x^{I+a}\partial y^{J+b}\partial t^{K+c}}\right)_j^n \frac{\Delta x^a}{a!}\frac{\Delta y^b}{b!}\frac{\Delta t^c}{c!} \tag{3.8}$$

where $C = I + J + K$, $A = N - C$, and $B = C + a + b + c$. Equation (3.7) is a special case of Eq. (3.8) with $A = N$ and $C = 0$.

Similarly, the fluxes $f_i^{x,y}$ and their derivatives inside a SE are also discretized by the Taylor series expansion:

$$\frac{\partial^C f_i^{x,y*}(x,y,t)}{\partial x^I \partial y^J \partial t^K} = \sum_{a=0}^{A} \sum_{b=0}^{A-a} \sum_{c=0}^{A-a-b} \left(\frac{\partial^B f_i^{x,y}}{\partial x^{I+a} \partial y^{J+b} \partial t^{K+c}} \right)_j^n \frac{\Delta x^a}{a!} \frac{\Delta y^b}{b!} \frac{\Delta t^c}{c!} \tag{3.9}$$

where C and B have the same definitions as that in Eq. (3.8). When $C = 0$, Eq. (3.9) is the Taylor series expansion of the flux $f_i^{x,y}$ itself. The coefficients of the Taylor series shown in Eqs. (3.8) and (3.9) are to be solved in the time marching calculation of a high-order CESE method.

3.2 Two-Dimensional Independent Variables

In this section, it is shown that not all the coefficients of the Taylor series shown in Eqs. (3.8) and (3.9) are the dependent variables in the context of the CESE method. Through the derivation it will be shown that the primary unknowns are the conserved variables $(u_i)_j^n$ and their spatial derivatives terms, e.g., $(u_{i,x})_j^n$, $(u_{i,y})_j^n$, $(u_{i,xy})_j^n, \ldots$, etc. All derivative terms of u_i involving temporal differentiation as well as all fluxes $(f^{x,y})_j^n$ and their spatial and temporal derivatives are not primary unknowns and are expressed as functions of the conserved variables and their spatial derivatives.

The first step is to express the fluxes and their derivatives as functions of the conserved variables and their derivatives. Two primary method to do this where used in the investigation. The first, listed below, uses the Jacobian and its derivatives and is not tied to any particular governing equations. This generality makes it ideal for this derivation but the method becomes more complex as the order of accuracy is increased. The other method, detailed in Chapter 5, directly evaluates the spatial and temporal derivatives of the fluxes without the use of a Jacobian matrix. This method is unique for each set of governing equations but is expandable to higher-orders with little effort. It is also more computationally efficient then the Jacobian method.

The Jacobian method begins by applying the chain rule, to the first derivatives of $f_i^{x,y}$

$$\frac{\partial f_i^{x,y}}{\partial \Psi_1} = \sum_{l=1}^{m} \frac{\partial f_i^{x,y}}{\partial u_l} \frac{\partial u_l}{\partial \Psi_1}, \tag{3.10}$$

where $\Psi_1 = x, y$, and t. On the right hand side of Eq. (3.10), $\partial f_i^{x,y}/\partial u_l$ is the Jacobian matrix, which is readily available for any two-dimensional system of equations. For the second derivatives,

$$\frac{\partial^2 f_i^{x,y}}{\partial \Psi_1 \partial \Psi_2} = \sum_{l=1}^{m} \frac{\partial f_i^{x,y}}{\partial u_l} \frac{\partial^2 u_l}{\partial \Psi_1 \partial \Psi_2} + \sum_{l=1}^{m} \sum_{p=1}^{m} \frac{\partial^2 f_i^{x,y}}{\partial u_l \partial u_p} \frac{\partial u_l}{\partial \Psi_1} \frac{\partial u_p}{\partial \Psi_2}, \tag{3.11}$$

where $(\Psi_1, \Psi_2) = (x, x), (y, y), (t, t), (x, y), (y, x), (x, t)$, and (y, t). As a part of the second term on the right hand side of Eq. (3.11), $\partial^2 f_i^{x,y}/\partial u_l \partial u_p$ is a three-dimensional matrix, which is readily derived from the governing equations. For the third derivatives,

$$
\begin{aligned}
\frac{\partial^3 f_i^{x,y}}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} =& \sum_{l=1}^{m} \frac{\partial f_i^{x,y}}{\partial u_l}\frac{\partial^3 u_l}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} + \\
& \sum_{l=1}^{m}\sum_{p=1}^{m} \frac{\partial^2 f_i^{x,y}}{\partial u_l \partial u_p}\left(\frac{\partial^2 u_l}{\partial\Psi_1\partial\Psi_2}\frac{\partial u_p}{\partial\Psi_3} + \frac{\partial^2 u_l}{\partial\Psi_1\partial\Psi_3}\frac{\partial u_p}{\partial\Psi_2} + \frac{\partial^2 u_l}{\partial\Psi_2\partial\Psi_3}\frac{\partial u_p}{\partial\Psi_1}\right) + \\
& \sum_{l=1}^{m}\sum_{p=1}^{m}\sum_{q=1}^{m} \frac{\partial^3 f_i^{x,y}}{\partial u_l \partial u_p \partial u_q}\frac{\partial u_l}{\partial\Psi_1}\frac{\partial u_p}{\partial\Psi_2}\frac{\partial u_q}{\partial\Psi_3},
\end{aligned}
\tag{3.12}
$$

where $(\Psi_1, \Psi_2, \Psi_3) = (x, x, x)$, (y, y, y), (t, t, t), (x, y, t), (y, x, t), (x, x, y), (x, y, x), (y, x, x), (y, y, x), (y, x, y), (x, y, y), (y, y, t), (x, x, t), (y, t, t), and (x, t, t). As a part of the last term on the right hand side of Eq. (3.12), $\partial^3 f_i^{x,y}/\partial u_l \partial u_p \partial u_q$ is a four-dimensional matrix, which is readily derived based in the definition of $f_i^{x,y}$ as functions of u_i. Aided by Eqs. (3.10-3.12), all spatial and temporal derivatives of $f_i^{x,y}$ are related to the derivatives of u_i.

The next step in this derivation is to demonstrated that all of temporal derivatives of u_i are functions of the conserved variables and their spatial derivatives. This will be accomplished by first relating the temporal derivatives to the spatial derivatives of $f_i^{x,y}$. Then, through the relations derived previously the derivatives of $f_i^{x,y}$ are related back to the spatial derivatives of u_i. This is procedure is known as the Cauchy-Kowalewski procedure. Essentially, the approach uses the governing equation and its derivatives to relate the temporal derivatives of u_i to the derivatives of $f_i^{x,y}$. First, the original governing equations:

$$\frac{\partial u_i}{\partial t} = -\frac{\partial f_i^x}{\partial x} - \frac{\partial f_i^y}{\partial y}, \quad i = 1, \ldots, m \tag{3.13}$$

is differentiated with respect to x, y and t yielding

$$\begin{aligned} \frac{\partial^2 u_i}{\partial x \partial t} &= -\frac{\partial^2 f_i^x}{\partial x \partial x} - \frac{\partial^2 f_i^y}{\partial y \partial x}, \\ \frac{\partial^2 u_i}{\partial y \partial t} &= -\frac{\partial^2 f_i^x}{\partial x \partial y} - \frac{\partial^2 f_i^y}{\partial y \partial y}, \end{aligned} \tag{3.14}$$

and

$$\begin{aligned} \frac{\partial^2 u_i}{\partial t^2} &= -\frac{\partial^2 f_i^x}{\partial x \partial t} - \frac{\partial^2 f_i^y}{\partial y \partial t} \\ &= -\frac{\partial}{\partial x} \sum_{l=1}^{m} \frac{\partial f_i^x}{\partial u_l} \frac{\partial u_l}{\partial t} - \frac{\partial}{\partial y} \sum_{l=1}^{m} \frac{\partial f_i^y}{\partial u_l} \frac{\partial u_l}{\partial t} \\ &= \frac{\partial}{\partial x} \sum_{l=1}^{m} \frac{\partial f_i^x}{\partial u_l} \left[\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right] + \frac{\partial}{\partial y} \sum_{l=1}^{m} \frac{\partial f_i^y}{\partial u_l} \left[\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right] \\ &= \sum_{l=1}^{m} \frac{\partial f_i^x}{\partial u_l} \left(\frac{\partial^2 f_l^x}{\partial x^2} + \frac{\partial^2 f_l^y}{\partial y \partial x} \right) + \sum_{l=1}^{m} \sum_{p=1}^{m} \frac{\partial^2 f_i^x}{\partial u_l \partial u_p} \frac{\partial u_l}{\partial x} \left(\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right) \\ &+ \sum_{l=1}^{m} \frac{\partial f_i^y}{\partial u_l} \left(\frac{\partial^2 f_l^x}{\partial x \partial y} + \frac{\partial^2 f_l^y}{\partial y^2} \right) + \sum_{l=1}^{m} \sum_{p=1}^{m} \frac{\partial^2 f_i^y}{\partial u_l \partial u_p} \frac{\partial u_l}{\partial y} \left(\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right) \end{aligned} \tag{3.15}$$

The third derivatives are found by applying $\partial^2/\partial x^2$, $\partial^2/\partial y^2$, and $\partial^2/\partial x \partial y$ to Eq. (3.13), which results in

$$
\begin{aligned}
\frac{\partial^3 u_i}{\partial x^2 \partial t} &= -\frac{\partial^3 f_i^x}{\partial x^3} - \frac{\partial^3 f_i^y}{\partial x^2 \partial y}, \\
\frac{\partial^3 u_i}{\partial y^2 \partial t} &= -\frac{\partial^3 f_i^x}{\partial y^2 \partial x} - \frac{\partial^3 f_i^y}{\partial y^3}, \\
\frac{\partial^3 u_i}{\partial x \partial y \partial t} &= -\frac{\partial^3 f_i^x}{\partial x^2 \partial y} - \frac{\partial^3 f_i^y}{\partial x \partial y^2}
\end{aligned}
\tag{3.16}
$$

Aided by Eq. (3.15) and the chain rule, the third derivatives of u_i containing two temporal derivatives are expressed as:

$$
\begin{aligned}
&\frac{\partial^3 u_i}{\partial t^2 \partial \Psi} = \\
&\quad \sum_{l=1}^{m} \frac{\partial f_i^x}{\partial u_l} \left(\frac{\partial^3 f_l^x}{\partial x \partial x \partial \Psi} + \frac{\partial^3 f_l^y}{\partial y \partial x \partial \Psi} \right) + \sum_{l=1}^{m} \sum_{p=1}^{m} \frac{\partial^2 f_i^x}{\partial u_l \partial u_p} \frac{\partial u_p}{\partial \Psi} \left(\frac{\partial^2 f_l^x}{\partial x \partial x} + \frac{\partial^2 f_l^y}{\partial y \partial x} \right) + \\
&\quad \sum_{l=1}^{m} \sum_{p=1}^{m} \frac{\partial^2 f_i^x}{\partial u_l \partial u_p} \left[\frac{\partial u_l}{\partial x} \left(\frac{\partial^2 f_l^x}{\partial x \partial \Psi} + \frac{\partial^2 f_l^y}{\partial y \partial \Psi} \right) + \frac{\partial^2 u_l}{\partial x \partial \Psi} \left(\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right) \right] + \\
&\quad \sum_{l=1}^{m} \sum_{p=1}^{m} \sum_{q=1}^{m} \frac{\partial^3 f_i^x}{\partial u_l \partial u_p \partial u_q} \frac{\partial u_l}{\partial x} \frac{\partial u_q}{\partial \Psi} \left(\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right) + \\
&\quad \sum_{l=1}^{m} \frac{\partial f_i^y}{\partial u_l} \left(\frac{\partial^3 f_l^x}{\partial x \partial y \partial \Psi} + \frac{\partial^3 f_l^y}{\partial y \partial y \partial \Psi} \right) + \sum_{l=1}^{m} \sum_{p=1}^{m} \frac{\partial^2 f_i^y}{\partial u_l \partial u_p} \frac{\partial u_p}{\partial \Psi} \left(\frac{\partial^2 f_l^x}{\partial x \partial y} + \frac{\partial^2 f_l^y}{\partial y \partial y} \right) + \\
&\quad \sum_{l=1}^{m} \sum_{p=1}^{m} \frac{\partial^2 f_i^y}{\partial u_l \partial u_p} \left[\frac{\partial u_l}{\partial y} \left(\frac{\partial^2 f_l^x}{\partial x \partial \Psi} + \frac{\partial^2 f_l^y}{\partial y \partial \Psi} \right) + \frac{\partial^2 u_l}{\partial y \partial \Psi} \left(\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right) \right] + \\
&\qquad \sum_{l=1}^{m} \sum_{p=1}^{m} \sum_{q=1}^{m} \frac{\partial^3 f_i^y}{\partial u_l \partial u_p \partial u_q} \frac{\partial u_l}{\partial y} \frac{\partial u_q}{\partial \Psi} \left(\frac{\partial f_l^x}{\partial x} + \frac{\partial f_l^y}{\partial y} \right),
\end{aligned}
\tag{3.17}
$$

where Ψ is either x or y. The third temporal derivative of a flux term is expressed as:

$$\begin{aligned}\frac{\partial^3 u_i}{\partial^3 t} &= -\frac{\partial}{\partial t^2}\left(\frac{\partial f_i^x}{\partial x}+\frac{\partial f_i^y}{\partial y}\right)\\ &= -\frac{\partial^3 f_i^x}{\partial x \partial t^2}-\frac{\partial^3 f_i^y}{\partial y \partial t^2}.\end{aligned} \tag{3.18}$$

The derivatives on the right hand side of Eq. (3.18) can be expressed as a function of spatial derivative terms only. Aided by Eq. (3.12), Eq. (3.18) is expressed as:

$$\begin{aligned}\frac{\partial^3 u_i}{\partial^3 t} =& -\sum_{l=1}^{m}\frac{\partial f_i^x}{\partial u_l}\underbrace{\frac{\partial^3 u_l}{\partial x \partial t^2}}_{\text{Eq.}(3.17)\Psi=x} - \frac{\partial f_i^y}{\partial u_l}\underbrace{\frac{\partial^3 u_l}{\partial y \partial t^2}}_{\text{Eq.}(3.17)\Psi=y}\\ &-\sum_{l=1}^{m}\sum_{p=1}^{m}\frac{\partial^2 f_i^x}{\partial u_l \partial u_p}\left(2\underbrace{\frac{\partial^2 u_l}{\partial x \partial t}}_{\text{Eq.}(3.14)}\underbrace{\frac{\partial u_p}{\partial t}}_{\text{Eq.}(3.13)}+\underbrace{\frac{\partial^2 u_l}{\partial t^2}}_{\text{Eq.}(3.15)}\frac{\partial u_p}{\partial x}\right)\\ &-\sum_{l=1}^{m}\sum_{p=1}^{m}\frac{\partial^2 f_i^y}{\partial u_l \partial u_p}\left(2\underbrace{\frac{\partial^2 u_l}{\partial y \partial t}}_{\text{Eq.}(3.14)}\underbrace{\frac{\partial u_p}{\partial t}}_{\text{Eq.}(3.13)}+\underbrace{\frac{\partial^2 u_l}{\partial t^2}}_{\text{Eq.}(3.15)}\frac{\partial u_p}{\partial y}\right)\\ &-\sum_{l=1}^{m}\sum_{p=1}^{m}\sum_{q=1}^{m}\left(\frac{\partial^3 f_i^x}{\partial u_l \partial u_p \partial u_q}\frac{\partial u_l}{\partial x}+\frac{\partial^3 f_i^y}{\partial u_l \partial u_p \partial u_q}\frac{\partial u_l}{\partial y}\right)\underbrace{\frac{\partial u_p}{\partial t}\frac{\partial u_q}{\partial t}}_{\text{Eq.}(3.13)}\end{aligned} \tag{3.19}$$

In Eq. (3.19), the temporal derivatives are replaced by the previously derived equations as indicated in the equation. This method is expandable to any desired order but as evident in the derivation each successive derivatives becomes more complex.

In general, the additional equations derived from the original model equations are cast into the following form:

$$\frac{\partial^C u_i}{\partial x^I \partial y^J \partial t^K} = -\frac{\partial^C f_i^x}{\partial x^{I+1} \partial y^J \partial t^{K-1}} - \frac{\partial^C f_i^y}{\partial x^I \partial y^{J+1} \partial t^{K-1}}, \tag{3.20}$$

where $C = I + J + K$, $K = 1, 2, \ldots, N$, $J = 0, 1, \ldots, N - K$, and $I = 0, 1, \ldots, N - K - J$. The procedure outline above is recursive and is extendable to higher derivatives of u_i.

To recap, all temporal and spatial derivatives of $f_i^{x,y}$ are expressed as functions of the conserved variables and its spatial and temporal derivatives. Further more all temporal derivatives of the conserved variables are explicit functions of the conserved variables and its spatial derivatives. This is achieved by using (i) the additional equations derived from the original model equations as shown in Eqs. (3.16-3.20), (ii) the chain rule and the Jacobian matrices as shown in Eqs. (3.10-3.12). As a results of this formulation the primary unknowns of a two-dimensional fourth-order CESE scheme are easily determined and shown in Table 3.1.

Table 3.1: Primary unknowns of the two-dimensional, fourth-order CESE method.

Even derivative variables	Odd derivative variables	
$(u_i)_j^n$	$(u_{i,x})_j^n$	$(u_{i,y})_j^n$
$(u_{i,xx})_j^n$	$(u_{i,xxx})_j^n$	$(u_{i,xxy})_j^n$
$(u_{i,xy})_j^n$	$(u_{i,xyx})_j^n$	$(u_{i,xyy})_j^n$
$(u_{i,yx})_j^n$	$(u_{i,yxx})_j^n$	$(u_{i,yxy})_j^n$
$(u_{i,yy})_j^n$	$(u_{i,yyx})_j^n$	$(u_{i,yyy})_j^n$

3.3 Two-Dimensional Space-Time Integration

In this section the original model equation, Eq. (3.3), is integrated over the CCE to calculate the primary unknown $(u_i)_j^n$ at the solution point. Due to the higher order polynomials in the Taylor series the integration over the CCE is more complex than the original second-order CESE method.

In the following discussion, the archetypal CCE and the solution point $3^\times$ as shown in Figs. 3.1 and 3.2 will be used. The integration over the CCE associated with point $3^\times$ includes the following three steps: (i) the space-time flux through the six side surfaces of the CCE, (ii) the flux through the bottom surface of the CCE, and (iii) the flux through the top surface of the CCE. The flux through the side and bottom surfaces are calculated from the known values at the previous time step in the neighboring cells. For example in Fig. (3.1) when updating cell 3 the solutions at $1'^\times$, $5'^\times$, $7'^\times$ are used to calculate the flux through the bottom and side surfaces. The apostrophe symbol denotes the previous time step and the symbol $\times$ denotes the solution point. Note that point $1'^\times$ is different from point $1'$. Point $1'^\times$ is the centroid of the hexagonal surface and point $1'$ is the centroid of the triangle. The calculation for fluxes through side and bottom surfaces are explicit. The flux through the top surface is a function of the solutions at the new time level. Even though the flux through the top surface uses the current solution the space-time integration remains explicit as long as the unknowns are calculated in the correct order.

3.3.1 Two-Dimensional Side Integration

The mesh stencil of the fourth-order CESE method is identical to the second-order CESE method [59]. Figure (3.3) shows the side plane surfaces associated with the solution point at $(x_{1^\times}, y_{1^\times}, t^{n-1/2})$.

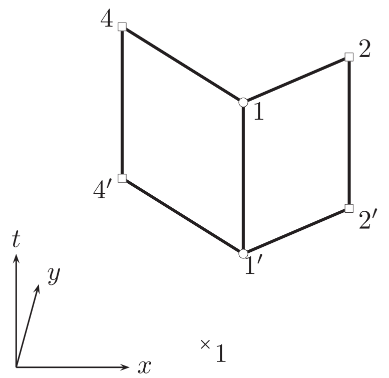


Figure 3.3: Two vertical side surfaces of the CCE associated with point $3^\times$

.

For a triangular mesh, there are six vertical surfaces as a part of the CCE associated with the solution point $3^\times$ located at $(x_{3^\times}, y_{3^\times}, t^n)$. These side surfaces are the exterior faces of the hexagonal box shown in Fig. (3.2b). On a side surface, the profiles of u_i, f_i^x and f_i^y are described by a Taylor series expansion with respect to the corresponding solution points, i.e., point $1'^\times$, $5'^\times$ and $7'^\times$ located at $(x_{1'^\times}, y_{1'^\times}, t^{n-1})$, $(x_{5'^\times}, y_{5'^\times}, t^{n-1})$, and $(x_{7'^\times}, y_{7'^\times}, t^{n-1})$, respectively.

Shown in Fig. (3.3), the following two rectangular surface $\square\ 1'2'21$ and $\square\ 1'4'41$ are considered. The integration of one side surface $\square\ 1'2'21$ by using the Taylor series expansion with respect to point $1'^{\times}$ located at $(x_{1'^{\times}}, y_{1'^{\times}}, t^{n-1/2})$ is used to demonstrate the procedure. In general the spatial location $(x_{1'^{\times}}, y_{1'^{\times}})$ does not coincide with $(x_{1'}, y_{1'})$. First the coefficients of the Taylor series representing the fluxes are calculated. Then the Taylor series is integrated over the side surface $\square\ 1'2'21$ to find the flux through the surface.

The space-time flux passing the side surface $\square\ 1'2'21$ is expressed as

$$(F_i)_{1'2'21} = \iint_{1'2'21} \mathbf{h}_i \cdot \mathbf{n} d\sigma \tag{3.21}$$

where the unit normal vector $\mathbf{n}$ pointing outward of CCE on the side surface $\square\ 1'2'21$ is

$$\mathbf{n} = \frac{[(y_{2'} - y_{1'}), -(x_{2'} - x_{1'}), 0]}{\sqrt{(x_{2'} - x_{1'})^2 + (y_{2'} - y_{1'})^2}}.$$

and the differential area for the integration is

$$d\sigma = dsdt,$$

$$ds = d\alpha\sqrt{(x_{2'} - x_{1'})^2 + (y_{2'} - y_{1'})^2},$$

$$dt = (\Delta t/2)d\beta.$$

A single parameter α to represent spatial displacement is possible because the projection of the side face $\square\ 1'2'21$ to the spatial domain is the line segment $\overline{12}$. Aided by these definitions Eq. (3.21) is rewritten as:

$$(F_i)_{1'2'21} = \frac{\Delta t}{2} \iint_{\alpha,\beta=0}^{1} \left[(y_{2'} - y_{1'})(f_i^x)^* - (x_{2'} - x_{1'})(f_i^y)^* \right] d\alpha d\beta. \tag{3.22}$$

The calculation in Eq. (3.22) depends on the known values of all the derivatives of $f_i^{x,y}$ at point $1'^{\times}$ in the previous time step. Through the Taylor series expansion, the value of $f_i^{x,y}$ at (x, y) on the side surface $\square\ 1'2'21$ is expressed as

$$\left(f_i^{x,y}\right)^* = \sum_{a=0}^{A} \sum_{b=0}^{A-a} \sum_{c=0}^{A-a-b} \left(\frac{\partial^{a+b+c} f_i^{x,y}}{\partial x^a \partial y^b \partial t^c} \right)_{1'^{\times}}^{n-1/2} \frac{\left(x - x_{1'^{\times}}\right)^a \left(y - y_{1'^{\times}}\right)^b \left(t - t^{n-1/2}\right)^c}{a!b!c!}. \tag{3.23}$$

The superscript $*$ denotes the discretized variables. The derivatives of the flux function, i.e., the coefficients of the polynomial, are stored at the solution point $1'^{\times}$ located at $(x_{1'^{\times}}, y_{1'^{\times}}, t^{n-1/2})$. Thus, the following parametric relations are generated for Eq. (3.23):

$$\begin{aligned} x &= \alpha(x_{2'} - x_{1'}) + x_{1'}, \\ y &= \alpha(y_{2'} - y_{1'}) + y_{1'}, \\ t - t^{n-1/2} &= \beta \frac{\Delta t}{2}, \end{aligned}$$

where $0 \leq \alpha \leq 1$, and $0 \leq \beta \leq 1$. This leads to the equation,

$$\sum_{x^i}^{x,y} \sum_{a=0}^{N} \sum_{b=0}^{N-a} \sum_{c=0}^{\substack{N- \\ a-b}} \frac{\partial^{a+b+c} f_i^{x^i}}{\partial x^a \partial y^b \partial t^c} \frac{N_{x^i}^{(2)}}{a!b!(c+1)!} \left(\frac{\Delta t}{2}\right)^{c+1}$$
$$\int_0^1 \left[(x_{2'} - x_{1'})\,\alpha + x_{1'} - x_{1^\times}\right]^a \left[(y_{2'} - y_{1'})\,\alpha + y_{1'} - y_{1^\times}\right]^b d\alpha, \tag{3.24}$$

For easy reference Eq. (3.24) is referenced as [Eq. (3.24)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_j, \Delta t, f_i^{x^i})$. Equation 3.24 is easily integrated by hand or though the use of a symbolic math application such as MATLAB. Listed below are a few sample integrations that are easily modified to provide all integrations necessary for a two-dimensional fourth order algorithm. To start let $\Delta x_i = x_i - x_j$, $\Delta y_i = y_i - y_j$ for $i = 1, 2$. With this definition the integral in Eq. (3.24) becomes

$$\mathcal{L}_{a,b}^{2s} = \int_0^1 \prod_{j=1}^{2} \left[\Delta\Psi_2^j \alpha + \Delta\Psi_1^j (1-\alpha)\right]^{\omega^j} d\alpha,$$

where $\Psi = \{x, y\}$ and $\omega = \{a, b\}$. When $a = 1$ and $b = 0$ the integration is

$$\mathcal{L}_{1,0,0}^{2s} = \frac{1}{2} \sum_{i=1}^{2} \Delta x_i.$$

When $a = b = 1$

$$\mathcal{L}_{1,1}^{2s} = \frac{1}{6} \left[\sum_{i=1}^{2} (\Delta x_i \Delta y_i) + \sum_{i=1}^{2} \Delta x_i \sum_{i=1}^{2} \Delta y_i\right].$$

When $a = 2$ and $b = 1$

$$\mathcal{L}^{2s}_{2,1} = \frac{1}{12}\Big[\sum_{i=1}^{2}\Delta x_i\sum_{i=1}^{2}\Delta y_i\sum_{i=1}^{2}\Delta x_i + 2\sum_{i=1}^{2}(\Delta x_i\Delta y_i\Delta x_i) + \sum_{i=1}^{2}(\Delta x_i\Delta y_i)\sum_{i=1}^{2}\Delta x_i + \sum_{i=1}^{2}(\Delta x_i\Delta x_i)\sum_{i=1}^{2}\Delta y_i + \sum_{i=1}^{2}(\Delta y_i\Delta x_i)\sum_{i=1}^{2}\Delta x_i\Big].$$

Given these three formulas it is possible to easily derive all other integrals for a fourth-order scheme. For example to obtain the formulation for $\mathcal{L}^{2s}_{2,0}$, $a = 2, b = 0$, one may substitute Δx_i for Δy_i in the formulation for $\mathcal{L}^{2s}_{1,1}$ resulting in

$$\mathcal{L}^{2s}_{1,1} = \frac{1}{6}\left[\sum_{i=1}^{2}(\Delta x_i\Delta x_i) + \sum_{i=1}^{2}\Delta x_i\sum_{i=1}^{2}\Delta x_i\right] = \frac{1}{6}\left[\sum_{i=1}^{2}(\Delta x_i)^2 + \left(\sum_{i=1}^{2}\Delta x_i\right)^2\right].$$

The calculation of fluxes through the other side surfaces is similar to that shown in Eq. (3.22). For a triangular mesh the flux through all six side surfaces of the CCE associated with the solution point $3^{\times}$ located at $(x_{3^\times}, y_{3^\times}, t^n)$ is readily calculated:

$$\begin{aligned}(F_i)_{\text{side}} =& (F_i)_{1'2'21} + (F_i)_{1'4'41} + (F_i)_{5'2'25} + \\ & (F_i)_{5'6'65} + (F_i)_{7'4'47} + (F_i)_{7'6'67}.\end{aligned} \tag{3.25}$$

3.3.2 Two-Dimensional Bottom and Top Integration

To proceed the calculation of the flux through the bottom surface of the CCE located at the time step $n - 1/2$ is presented. For a triangular mesh the bottom surface is a hexagon defined by the vertices: $1'2'5'6'7'4'$ listed in Fig. (3.2b). This hexagon is divided into into 3 BCEs: $\square\ 1'2'3'4'$, $\square\ 5'6'3'2'$, and $\square\ 7'4'3'6'$. The flux through $\square\ 1'2'3'4'$, is calculated based on the solutions stored at point $1'^{\times}$ located at $(x_{1'^{\times}}, y_{1'^{\times}}, t^{n-1/2})$. The flux through $\square\ 5'6'3'2'$, is calculated based on the solutions stored at point $5'^{\times}$ located at $(x_{5'^{\times}}, y_{5'^{\times}}, t^{n-1/2})$. The flux through $\square\ 7'4'3'6'$, is calculated based on the solutions stored at point $7'^{\times}$ located at $(x_{7'^{\times}}, y_{7'^{\times}}, t^{n-1/2})$.

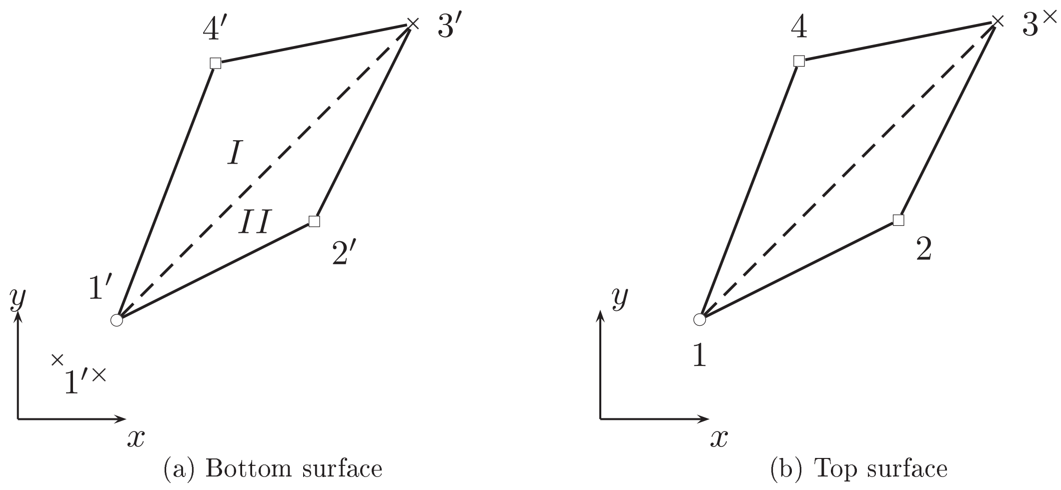


Figure 3.4: A part of the bottom and the top surfaces of the CCE.

To proceed the quadrilateral $\square\ 1'2'3'4'$, shown in Fig. (3.4a), is split into two triangles $\Delta 1'3'4'$ and $\Delta 1'2'3'$ in order to facilitate the integration. Since the procedures of calculating the flux through each of the triangles of the three quadrilaterals are identical, only the space-time flux integration over $\Delta 1'3'4'$ is shown:

$$(F_i)_{1'3'4'} = \iint_{1'3'4'} \mathbf{h}_i \cdot \mathbf{n} d\sigma \tag{3.26}$$

Here, the unit normal vector pointing outward of the CCE on the bottom surface of the CCE is $\mathbf{n} = (0, 0, -1)$. Aided by the orientation of $\mathbf{n}$ and the definition of $\mathbf{h}_i$, Eq. (3.26) becomes

$$(F_i)_{1'3'4'} = -\iint_{1'2'3'4'} (u_i)^* d\sigma \tag{3.27}$$

where $(u_i)^*$ is the discretized u_i profile over $\Delta 1'3'4'$ by using the third-order Taylor series:

$$(u_i)^* = \sum_{a=0}^{A} \sum_{b=0}^{A-a} \frac{1}{a!b!} \left(\frac{\partial^{a+b} u_i}{\partial x^a \partial y^b} \right)_{1'\times}^{n-1/2} (x - x_{1'\times})^a \, (y - y_{1'\times})^b \tag{3.28}$$

The temporal derivatives are not needed in this Taylor series because all points on $\Delta 1'3'4'$ are at the same time step, $t = t^{n-1/2}$.

For computational efficiency, the integral is analytically derived by applying coordinate transformation to transform $\Delta 1'3'4'$ into a right triangle. Let the new coordinates be (ξ, η). The coordinate transformation from (x, y) to (ξ, η) is such that at point $3'$, $(\xi, \eta) = (1, 0)$, at point $1'$, $(\xi, \eta) = (0, 0)$, and at point $4'$, $(\xi, \eta) = (0, 1)$, Thus, the transformation equation is

$$\begin{aligned}\begin{pmatrix} x - x_{1'} \\ \\ y - y_{1'} \end{pmatrix} &= \begin{pmatrix} x_{3'} - x_{1'} & x_{4'} - x_{1'} \\ \\ y_{3'} - y_{1'} & y_{4'} - y_{1'} \end{pmatrix} \begin{pmatrix} \xi \\ \\ \eta \end{pmatrix} \\ &= \begin{pmatrix} x_\xi & x_\eta \\ \\ y_\xi & y_\eta \end{pmatrix} \begin{pmatrix} \xi \\ \\ \eta \end{pmatrix} \\ &= J \begin{pmatrix} \xi \\ \\ \eta \end{pmatrix}\end{aligned} \tag{3.29}$$

where $0 \le \xi \le 1$, $0 \le \eta \le 1 - \xi$, $x_\xi = x_{3'} - x_{1'}$, $x_\eta = x_{4'} - x_{1'}$, $y_\xi = y_{3'} - y_{1'}$, $y_\eta = y_{4'} - y_{1'}$, J is the Jacobian matrix of the coordinate transformation. x and y in Eq. (3.29) can now be expressed in terms of ξ, η via:

$$\begin{pmatrix} x - x_{1'^\times} \\ \\ y - y_{1'^\times} \end{pmatrix} = J \begin{pmatrix} \xi \\ \\ \eta \end{pmatrix} + \begin{pmatrix} x_{1'} - x_{1'^\times} \\ \\ y_{1'} - y_{1'^\times} \end{pmatrix}.$$

Point $1'^\times$ is the solution point, where the Taylor series is expanded from. Point $1'$ on the other hand is the vertex of the triangle $\Delta 1'3'4'$. In general, point $1' \neq$ point $1'^\times$. The determinant of the Jacobian matrix J is

$$\begin{aligned}\det(J) = &= x_\xi y_\eta - y_\eta x_\xi \\ &= (x_{3'} - x_{1'})(y_{4'} - y_{1'}) - (x_{4'} - x_{1'})(y_{3'} - y_{1'}).\end{aligned}$$

The absolute value of $\det(J)$ is twice the size of the triangle $\Delta 1'3'4'$.

Aided by the coordinate transformation and Eq. (3.28) the space-time flux shown in Eq. (3.27) is expressed as

$$\begin{aligned}(F_i)_{1'3'4'} = -\sum_{a=0}^{A}\sum_{b=0}^{A-a}\frac{1}{a!b!}\left(\frac{\partial^{a+b}u_i}{\partial x^a \partial y^b}\right)_{1'^{\times}}^{n-1/2} \det(J) \\ \int_0^1\int_0^{1-\xi}\left(x_\xi\xi + x_\eta\eta + \Delta x_{1^\times}\right)^a\left(y_\xi\xi + y_\eta\eta + \Delta y_{1^\times}\right)^b d\xi d\eta.\end{aligned} \tag{3.30}$$

where $\Delta x_{1^\times} = x_{1'} - x_{1'^\times}$ and $\Delta y_{1^\times} = y_{1'} - y_{1'^\times}$. For ease of reference Eq. (3.30) is expressed as [Eq. (3.30)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_j, u_{jcl})$. The integral of the polynomial in Eq. (3.30) for all combinations of a and b are readily derived. In the present study a symbolic mathematics package is used to integrate Eq. (3.30).

In the equations that follow a few select integrals for the integral in Eq. (4.19) will be evaluated. The integrals provided are all that are required by a fourth-order space-time integration procedure. To start let $\Delta x_i = x_i - x_{j^\times}$, $\Delta y_i = y_i - y_{j^\times}$, for $i = 1, 2, 3$. Note that the values of i are not related to the points $\{1, 2, 3\}$ in Fig. (3.1). Instead they represent the three vertices on a triangular surface. With these definitions Eq. (4.19) is rewritten as:

$$\mathcal{L}_{a,b,c}^{2bt} = \int_0^1\int_0^{1-\eta_k}\prod_{j=1}^{2}\left[\Delta\Psi_2^j\eta_k + \Delta\Psi_3^j\xi_k + \Delta\Psi_1^j\left(1-\eta_k-\xi_k\right)\right]^{\omega^j} d\xi_k d\eta_k,$$

where $\Psi = \{x, y\}$ and $\omega = \{a, b\}$. When $a = 1$ and $b = 0$ the integral is evaluated as

$$\mathcal{L}^{2bt}_{1,0,0} = \frac{1}{6}\sum_{i=1}^{3}\Delta x_i.$$

When $a = b = 1$

$$\mathcal{L}^{2bt}_{1,1,0} = \frac{1}{24}\left[\sum_{i=1}^{3}(\Delta x_i \Delta y_i) + \sum_{i=1}^{3}\Delta x_i \sum_{i=1}^{3}\Delta y_i\right].$$

When $a = 2$ and $b = 1$

$$\mathcal{L}^{2bt}_{2,1} = \frac{1}{60}\Big[\sum_{i=1}^{3}\Delta x_i \sum_{i=1}^{3}\Delta y_i \sum_{i=1}^{3}\Delta x_i + 2\sum_{i=1}^{3}(\Delta x_i \Delta y_i \Delta x_i) + \\ \sum_{i=1}^{3}(\Delta x_i \Delta y_i)\sum_{i=1}^{3}\Delta x_i + \sum_{i=1}^{3}(\Delta x_i \Delta x_i)\sum_{i=1}^{3}\Delta y_i + \sum_{i=1}^{3}(\Delta y_i \Delta x_i)\sum_{i=1}^{3}\Delta x_i\Big].$$

As with the two-dimensional side face derivation the other integrals required for a fourth-order scheme are easily obtained through substituting. For example $\mathcal{L}^{2bt}_{2,0}$ is obtained by substituting Δx_i for Δy_i in the formulation of $\mathcal{L}^{2bt}_{1,1}$.

The fluxes through $\Delta 1'2'3'$ as well as through the other sub triangles in quadrilaterals associated with points $5'$ and $7'$ are calculated in a similar manner. Thus the space-time flux through the whole bottom surface of the CCE associated with point

$3^\times$ is expressed as

$$
\begin{aligned}
(F_i)_{\text{bot}} =& (F_i)_{\Delta 1'3'4'} + (F_i)_{\Delta 1'2'3'} + \\
& (F_i)_{\Delta 5'6'3'} + (F_i)_{\Delta 5'3'2'} + \\
& (F_i)_{\Delta 7'3'6'} + (F_i)_{\Delta 7'4'3'}.
\end{aligned} \tag{3.31}
$$

The calculation of the space-time flux through the top surface of the CCE associated with point $3^\times$ is similar to the flux calculation through the bottom surface of the CCE. The only difference is that there is only one solution point, $3^\times$, for flux on the top surface while the calculation of the flux through the bottom surface must use the solutions on three solution points, i.e., points $1'^\times, 5'^\times$, and $7'^\times$. To proceed, the hexagonal area of the top surface of the CCE is divided into three quadrilaterals: $\square 1234, \square 5632$, and $\square 7436$. Then each quadrilateral is further divided into two triangles: $\square 1234 = \Delta 341 + \Delta 312$, $\square 5632 = \Delta 328 + \Delta 356$, and $\square 7436 = \Delta 367 + \Delta 374$. As such, the calculation of the flux through the top surface of the CCE is divided into the flux through six triangles. In what follows, the calculation of flux through $\Delta 134$ is illustrated.

$$
(F_i)_{\Delta 134} = \iint_{\Delta 134} \mathbf{h}_i \cdot \mathbf{n} dR \tag{3.32}
$$

where $\mathbf{n} = (0, 0, 1)$, and $\mathbf{h}_i \cdot \mathbf{n} = u_i$. To proceed, the coordinates are transform from

(x, y) to (ξ, η) and have

$$\begin{aligned}(F_i)_{134} =& \int_{\xi,\eta} (u_i)^* \det(J) d\xi d\eta \\ =& \sum_{a=0}^{A} \sum_{b=0}^{A-a} \frac{1}{a!b!} \left(\frac{\partial^{a+b} u_i}{\partial x^a \partial y^b} \right)^n_{3^\times} \det(J) \\ & \int_0^1 \int_0^{1-\xi} \left(\xi x_\xi + \eta x_\eta + \Delta x_3 \right)^a \left(\xi y_\xi + \eta y_\eta + \Delta y_3 \right)^b d\eta d\xi.\end{aligned} \tag{3.33}$$

where $\Delta x_3 = x_3 - x_{3^\times}$ and $\Delta y_3 = y_3 - y_{3^\times}$ and A is the order of the polynomial, $A = 3$ for a fourth-order CESE scheme. Since this equation will be referred to later it is convenient to express it as: [Eq. (3.33)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_j, u_{icl})$. Similar to the bottom surface, the total flux through the top surface is

$$\begin{aligned}(F_i)_{\text{top}} =& (F_i)_{134} + (F_i)_{123} + \\ & (F_i)_{563} + (F_i)_{532} + \\ & (F_i)_{736} + (F_i)_{743}.\end{aligned} \tag{3.34}$$

Equation (3.34) is an implicit function of the conserved variable u_i and its spatial derivatives stored at point $3^\times$. All the derivatives are evaluated at Point $3^\times$ and are moved outside of the integration as shown in Eq. (3.33). As a result, the integration in Eq. (3.33) is readily obtained by from mesh geometry. For efficiency this information may also be stored for future use. It is evident from this equation that all of the unknowns for the current time step are included in this equations. Due to the definition of the CE the integral the first derivatives are null. The values of the first derivative are non-zero the integral over the entire CE is zero. Furthermore if the derivatives are determined in the correct order all other Taylor series coefficients will have been calculated thus making the space-time integration procedure explicit. This situation is identical to that in the original second-order CESE method. The final step requires summing the over all surfaces and solving for $(u_i)_{3^\times}$ through this equation an explicit formulation to progress the conserved variables is formulated.

$$(u_i)_{3^\times} = \frac{-1}{A_{\text{hex}}} \left[(F_i)_{\text{side}} + (F_i)_{\text{bot}} + (F_i)_{\text{top}} \right] \tag{3.35}$$

where $(F_i)_{\text{top}}$ denotes the part of the space-time flux in Eq. (3.34), which is calculated from the known second, third, and other higher derivatives of u_i stored at point $3^\times$.

3.3.3 Two-Dimensional Even Derivative Terms

This method is also used to calculate the even derivatives of the conserved variables. Since this procedure is easily expanded to solve any even derivatives for any order scheme, the second derivatives in the setting of a fourth-order scheme will be used as an example. For example in a fourth-order scheme the second derivatives listed in Table 3.1 would also be calculated using the space-time integration procedure. Essentially, the integration procedure is applied four times to solve for: $(u_{i,xx})^n_j$, $(u_{i,xy})^n_j$, $(u_{i,yx})^n_j$, $(u_{i,yy})^n_j$, at all the solution points in the new time level.

To proceed, Eq. (3.16) is cast into into the following divergence-free equations:

$$\begin{aligned} &\nabla \cdot \mathbf{h}_{i,xx} = 0, \quad \nabla \cdot \mathbf{h}_{i,xy} = 0, \\ &\nabla \cdot \mathbf{h}_{i,yy} = 0, \quad \nabla \cdot \mathbf{h}_{i,yx} = 0. \end{aligned} \tag{3.36}$$

where

$$\begin{aligned} &\mathbf{h}_{i,xx} = (f^x_{i,xx}, f^y_{i,xx}, u_{i,xx}), \quad \mathbf{h}_{i,xy} = (f^x_{i,xy}, f^y_{i,xy}, u_{i,xy}), \\ &\mathbf{h}_{i,yy} = (f^x_{i,yy}, f^y_{i,yy}, u_{i,yy}), \quad \mathbf{h}_{i,yx} = (f^x_{i,yx}, f^y_{i,yx}, u_{i,yx}) \end{aligned} \tag{3.37}$$

are the additional space-time flux vectors. It should be noted that both $\mathbf{h}_{i,xy}$ and $\mathbf{h}_{i,yx}$ appear in Eq. (3.36) because $\frac{\partial^2 u}{\partial x \partial y}$ and $\frac{\partial^2 u}{\partial y \partial x}$ are treated as two distinct variables.

Aided by the Gauss theorem in the three-dimensional space-time domain, the above differential equations are recast into the following integral equations:

$$\begin{aligned} &\oint \mathbf{h}_{i,xx} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,xy} \cdot d\mathbf{s} = 0, \\ &\oint \mathbf{h}_{i,yy} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,yx} \cdot d\mathbf{s} = 0 \end{aligned} \tag{3.38}$$

The space-time integration procedure is applied four times to enforce flux conservation of $\mathbf{h}_{i,xx}$, $\mathbf{h}_{i,xy}$, $\mathbf{h}_{i,yy}$, and $\mathbf{h}_{i,yx}$. In the context of a fourth-fourth order scheme each of the second derivatives is expressed as a first-order Taylor series:

$$\begin{aligned}
u^*_{i,xx} &= u_{i,xx} + u_{i,xxx}\Delta x + u_{i,xxy}\Delta y + u_{i,xxt}\Delta t \\
u^*_{i,xy} &= u_{i,xy} + u_{i,xyx}\Delta x + u_{i,xyy}\Delta y + u_{i,xyt}\Delta t \\
u^*_{i,yx} &= u_{i,yx} + u_{i,yxx}\Delta x + u_{i,yxy}\Delta y + u_{i,yxt}\Delta t \\
u^*_{i,yy} &= u_{i,yy} + u_{i,yyx}\Delta x + u_{i,yyy}\Delta y + u_{i,yyt}\Delta t
\end{aligned} \tag{3.39}$$

The associated fluxes are also expressed as a first-order Taylor series:

$$\begin{aligned}
(f^{x,y}_{i,xx})^* &= f^{x,y}_{i,xx} + f^{x,y}_{i,xxx}\Delta x + f^{x,y}_{i,xxy}\Delta y + f^{x,y}_{i,xxt}\Delta t \\
(f^{x,y}_{i,xy})^* &= f^{x,y}_{i,xy} + f^{x,y}_{i,xyx}\Delta x + f^{x,y}_{i,xyy}\Delta y + f^{x,y}_{i,xyt}\Delta t \\
(f^{x,y}_{i,yx})^* &= f^{x,y}_{i,yx} + f^{x,y}_{i,yxx}\Delta x + f^{x,y}_{i,yxy}\Delta y + f^{x,y}_{i,yxt}\Delta t \\
(f^{x,y}_{i,yy})^* &= f^{x,y}_{i,yy} + f^{x,y}_{i,yyx}\Delta x + f^{x,y}_{i,yyy}\Delta y + f^{x,y}_{i,yyt}\Delta t
\end{aligned} \tag{3.40}$$

Since these are linear equations the space-time integration procedure is nearly identical to original second-order scheme developed by Wang and Chang[59]. Furthermore, when applying these equations to the top surface as derived in Section 3.3.2 the only non-zero integration term is the second derivatives. This means that the resulting integration has only one unknown and is solved explicitly. As an example the resulting integration for each surface is provided for $\mathbf{h}_{i,xx}$. To begin Eq. (3.24) takes on the form

$$
\begin{aligned}
(F_{i,xx})_{1'2'21} = \sum_{x^i}^{x,y} N_{x^i}^{(2)} \frac{\Delta t}{2} \Bigg(& f'^{x^i}_{i,xxx} \int_0^1 \left[(x_{2'} - x_{1'})\,\alpha + x_{1'} - x_{1^\times} \right] d\alpha + \\
& f'^{x^i}_{i,xxy} \int_0^1 \left[(y_{2'} - y_{1'})\,\alpha + y_{1'} - y_{1^\times} \right] d\alpha + \\
& 0.5 f'^{x^i}_{i,xxy} \frac{\Delta t}{2} + f'^{x^i}_{i,xx} \Bigg)
\end{aligned}
\tag{3.41}
$$

$$
\begin{aligned}
(F_{i,xx})_{1'3'4'} = -\det(J) \Bigg(& 0.5 u'_{i,xx} + \\
& u'_{i,xxx} \int_0^1 \int_0^{1-\xi} \left(x_\xi \xi + x_\eta \eta + \Delta x_{1^\times} \right) d\xi d\eta + \\
& u'_{i,xxy} \int_0^1 \int_0^{1-\xi} \left(y_\xi \xi + y_\eta \eta + \Delta y_{1^\times} \right) d\xi d\eta \Bigg).
\end{aligned}
\tag{3.42}
$$

$$
\begin{aligned}
(F_{i,xx})_{1'3'4'} = \det(J) \Bigg(& 0.5 u_{i,xx} + \\
& u_{i,xxx} \int_0^1 \int_0^{1-\xi} \left(x_\xi \xi + x_\eta \eta + \Delta x_{1^\times} \right) d\xi d\eta + \\
& u_{i,xxy} \int_0^1 \int_0^{1-\xi} \left(y_\xi \xi + y_\eta \eta + \Delta y_{1^\times} \right) d\xi d\eta \Bigg).
\end{aligned}
\tag{3.43}
$$

When the flux is summed over all surfaces an equation to progress $u_{i,xx}$ is obtained and has the form

$$
(u_i)_{3^\times} = \frac{-1}{A_{\text{CCE}}} \left[(F_i)_{\text{side}} + (F_i)_{\text{bot}} \right], \tag{3.44}
$$

It should be noted that the $(F_i)_{\text{top}}$ is absent from Eq. (3.44), which means that the evaluation of $u_{i,xx}$ is completely explicit. This same procedure is duplicated for $u_{i,xy}$, $u_{i,yx}$, and $u_{i,yy}$. This same procedure is easily applied to any higher derivatives.

3.4 Two-Dimensional Central Differencing Procedure

In this section the central differencing procedure used to find the odd derivatives at the new time step is illustrated. There are two primary methods to calculate the odd derivatives, the CFL insensitive $c - \tau$ scheme[59, 61] and the edge-based derivative (EBD) scheme [44]. Both method use a similar approach only differing in how they determine the locations to evaluate the Taylor series expansion at. This section is split into two subsections. The first derives an arbitrary order $c - \tau$ scheme, the second derives the arbitrary order EBD scheme. Both of these sections begin by deriving the core scheme which calculates the first derivatives and then follows up with a generalization to higher derivatives.

3.4.1 Two-Dimensional $c - \tau$ scheme

The $c - \tau$ scheme[61] is an extension of the original c scheme proposed by Wang and Chang[59]. The motivation behind the development of the $c - \tau$ scheme was to decrease the dissipation that occurred when the CFL number dropped below $\approx$ 0.1. The basic principles of this scheme is to take a Taylor series expansion from the current solution point to a location at the new time step. When this Taylor series is written out it is evident that some of the Taylor series expansions terms are unknown. For a second-order CESE scheme the only known values at this time step is u_i. The first derivatives, $u_{i,x}$ and $u_{i,y}$ are unknown, combine this with the value of u_i^* at the expansion point results in three unknown values. By performing multiple Taylor series expansions to other points a system of equations is generated that allows for the unknown values to be calculated. The rest of this subsection provides a complete derivation required to find the first derivatives of the unknowns using the $c-\tau$ scheme.

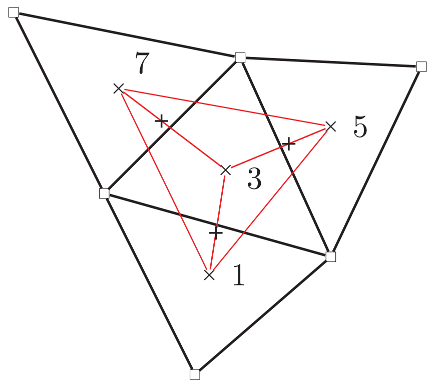


Figure 3.5: The stencils used for the odd derivative c-τ scheme.

The first step is to determine the location of the Taylor series expansion points, 1^+, 5^+ and 7^+, shown in Fig. (3.5). These points are calculated according to the $c-\tau$ scheme detailed by Chang and Wang [61]. In short this τ parameter shifts the expansion point towards the solution point, $3^{\times}$, as the CFL number decreases. One example function is

$$x_{j^+} = 0.5(x_{3^\times} + x_j) + 0.5(x_j - x_{3^\times})CFL$$

When the CFL number is 1 the $x_{j^+} = x_j$ and when the CFL number is zero x_{j^+} is equal to the average value of x_{j^+} and x_j. It should be noted that the points 1^+, 5^+ and 7^+ are located at the new time step. In the following derivation, the superscripts and subscripts outside of a bracket denote the location, from where the Taylor series are expanded and the coefficients are calculated.

Next, the Taylor series is expanded from the solution point $3^\times$ in the new time level to points 1^+, 5^+, and 7^+ at the same time step:

$$\left[(u_i)^n_{j^+}\right]^n_{3^\times} = \sum_{a=0}^{A}\sum_{b=0}^{A-a}\left(\frac{\partial^{a+b}u_i}{\partial x^a \partial y^b}\right)^n_{3^\times}\frac{\left(x_{j^+} - x_{3^\times}\right)^a\left(y_{j^+} - y_{3^\times}\right)^b}{a!b!}. \tag{3.45}$$

where $j = \{1,5,7\}$, and A is the equal to the order of the Taylor series expansion. For a fourth-order CESE scheme $A = 3$. Since the Taylor series in Eq. (3.45) uses the values at the new time step there could be several unknown values on the RHS. In application this is not the case as long as the primary variables are calculated in the correct order. For the sake of the derivation it is assumed that all of the primary unknowns with the exception of $u_{i,x}$ and $u_{i,y}$ are known.

Next, let $[(u_i)^n_{j^+}]^{n-1/2}_{j^\times}$ be the Taylor series expansion from $j^\times$ at the previous half time step, $t = t^{n-1/2}$, to j^+ at the current time step $t = t^n$, where $j = \{1, 5, 7\}$. This Taylor series is expressed as:

$$\left[(u_i)^n_{j^+}\right]^{n-1/2}_{j^\times} = \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial t^c}\right)^{n-\frac{1}{2}}_{j^\times} \frac{\left(x_{j^+} - x_{j^\times}\right)^a \left(y_{j^+} - y_{j^\times}\right)^b}{a!b!c!}\left(\frac{\Delta t}{2}\right)^c \tag{3.46}$$

Next, let Eq. (3.46) be equal to Eq. (3.45), i.e., $[(u_i)^n_{j^+}]^n_{3^\times} = [(u_i)^n_{j^+}]^{n-1/2}_{j^\times}$ and move the first derivatives to the left hand side of the equation to yield

$$\begin{aligned}
&(u_{i,x})^n_{3^\times}\Delta x_{j^+} + (u_{i,y})^n_{3^\times}\Delta y_{j^+} \\
&= \left[(u_i)^n_{j^+}\right]^{n-1/2}_{j^\times} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\left(\frac{\partial^{a+b}u_i}{\partial x^a \partial y^b}\right)^n_{3^\times}\frac{\left(\Delta x_{j^+}\right)^a\left(\Delta y_{j^+}\right)^b}{a!b!},
\end{aligned} \tag{3.47}$$

where $\Delta x_{j^+} = x_{j^+} - x_{3^\times}$, $\Delta y_{j^+} = y_{j^+} - y_{3^\times}$ and $(a, b) \neq \big\{(1, 0), (0, 1)\big\}$. Applying Eq. (3.47) to points $\{1, 5, 7\}$ yields three algebraic equations with two unknowns, $(u_{i,x})^n_{3^\times}$ and $(u_{i,y})^n_{3^\times}$. The next step is to group each set of equations into pairs, i.e., $(1, 5)$, $(1, 7)$, or $(5, 7)$. By grouping these equations together a system of linear equations is formed. For example, by choosing points $(1, 5)$ the following two coupled equations for the unknowns $(u_{i,x})^n_{3^\times}$ and $(u_{i,y})^n_{3^\times}$ is formed:

$$\begin{bmatrix} \Delta x_{1^+} & \Delta y_{1^+} \\ \Delta x_{5^+} & \Delta y_{5^+} \end{bmatrix} \begin{bmatrix} (u_{i,x})^n_{3^\times} \\ (u_{i,y})^n_{3^\times} \end{bmatrix}^{(1,5)} = \begin{bmatrix} \left[(u_i)^n_{1^+}\right]^{n-1/2}_{1^\times} - \sum_{a=0}^{A} \sum_{b=0}^{A-a} \left(\frac{\partial^{a+b} u_i}{\partial x^a \partial y^b}\right)^n_{3^\times} \frac{\left(\Delta x_{1^+}\right)^a \left(\Delta y_{1^+}\right)^b}{a!b!} \\ \left[(u_i)^n_{5^+}\right]^{n-1/2}_{5^\times} - \sum_{a=0}^{A} \sum_{b=0}^{A-a} \left(\frac{\partial^{a+b} u_i}{\partial x^a \partial y^b}\right)^n_{3^\times} \frac{\left(\Delta x_{5^+}\right)^a \left(\Delta y_{5^+}\right)^b}{a!b!} \end{bmatrix} \tag{3.48}$$

where $(a, b) \neq (0, 1), (1, 0)$. This is repeated for the other pairs which results in the following possible solutions to $(u_{i,x})^n_{3^\times}$ and $(u_{i,y})^n_{3^\times}$:

$$\begin{bmatrix} (u_{i,x})^n_{3^\times} \\ (u_{i,y})^n_{3^\times} \end{bmatrix}^{(1,7)} \quad \text{and} \quad \begin{bmatrix} (u_{i,x})^n_{3^\times} \\ (u_{i,y})^n_{3^\times} \end{bmatrix}^{(5,7)}$$

The last step of the computation is to perform a reweighing algorithm to filter three sets of calculated solution in order obtain a solution of first derivatives capable of capturing shock waves. The weighting scheme is

$$u_{x^\times} = \frac{\sum_{r=1}^{N} \omega_r u_x^{(r)}}{\sum_{r=1}^{N} \omega_r} \qquad u_{y^\times} = \frac{\sum_{r=1}^{N} \omega_r u_y^{(r)}}{\sum_{r=1}^{N} \omega_r}$$

where N is the number of possible solutions, the superscript (r) represents one of the possible solutions and ω is the weight associated with each solution. The previously derived reweighing schemes from [61] and [72] are applicable to the high-order CESE method without modification. In this investigation a modified version of the S2 method from [72] and the reweighing scheme used in [63] was deployed. To begin let,

$$w_r = 1 + \sigma \sqrt{\left| \frac{\theta_{i_{max}}}{\theta_i^{(r)}} - 1 \right|}, \tag{3.49}$$

where

$$\theta_i^{(r)} = \sqrt{\left(u_{i,x}^{(r)}\right)^2 + \left(u_{i,y}^{(r)}\right)^2 + \epsilon}, \quad \theta_{i_{max}} = Max(\theta_i^{(1)}, \theta_i^{(2)} \dots),$$
$$\sigma = \sigma_0 / CFL,$$

where ϵ is a small positive number used to prevent dividing by zero. Theoretical the absolute value used in Eq. (3.49) is not required but in practice it is possible for $\frac{\theta_{i_{max}}}{\theta_i^{(r)}} - 1$ to be less than one, which would cause the square root operator to produce a NaN error.

By comparing Eq. (3.48) with the original scheme found in [59] it is evident that the only difference between the two equations is that a higher-order Taylor series is used in the new scheme. This method is easily altered to find any odd derivative. For example to find the variables $u_{i,xyx}$ and $u_{i,xyy}$ consider the Taylor series expansion of

$u_{i,xy}$ from $3^{\times}$ to j^{+}

$$u^{*}_{i,xy} = \left[u_{i,xy} + u_{i,xyx}\Delta x_{j^+} + u_{i,xyy}\Delta y_{j^+}\right]^{n}_{3^\times} + \sum_{a=0}^{A}\sum_{b=0}^{A-a}\left(\frac{\partial^{2+a+b}u_i}{\partial x^{a+1}\partial y^{b+1}}\right)^{n}_{3^\times}\frac{\Delta x^a}{a!}\frac{\Delta y^b}{b!}$$
$$\text{for } a,b \neq \left\{(0,0),(1,0),(0,1)\right\}. \tag{3.50}$$

The first three Taylor series coefficients are listed explicitly. This is important because the alternative rule for differentiation is not used and this will decrease the ambiguity for the terms listed in the summation.

An expression for the Taylor series expansion from the previous time step to the current one is:

$$\left[\left(u_{i,xy}\right)^{n}_{j^+}\right]^{n-1/2}_{j^\times} = \left[u_{i,xy} + u_{i,xyx}\left(x_{j^+} - x_{j^\times}\right) + u_{i,xyy}\left(y_{j^+} - y_{j^\times}\right) + u_{i,xyt}\frac{\Delta t}{2}\right]^{n-1/2}_{j^\times} +$$
$$\sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{2+a+b+c}u_i}{\partial x^{a+1}\partial y^{b+1}\partial t^{c}}\right)^{n-1/2}_{j^\times}\frac{\left(x_{j^+} - x_{j^\times}\right)^a}{a!}\frac{\left(y_{j^+} - y_{j^\times}\right)^b}{b!}\frac{\Delta t^c}{2c!}$$
$$\text{for } (a,b,c) \neq \left\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\right\}. \tag{3.51}$$

By applying the same procedure to $u^{*}_{i,xy}$ that was used for u^{*}_{i} a solution for $u_{i,xyx}$ and $u_{i,xyy}$ is derived:

$$
\begin{bmatrix} \Delta x_{1+} & \Delta y_{1+} \\ \\ \Delta x_{5+} & \Delta y_{5+} \end{bmatrix}
\begin{bmatrix} (u_{i,xyx})^n_{3\times} \\ \\ (u_{i,xyy})^n_{3\times} \end{bmatrix}^{(1,5)} =
$$
$$
\begin{bmatrix}
\left[(u_{i,xy})^n_{1+}\right]^{n-1/2}_{1\times} - (u_{i,xy})^n_{3\times} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\left(\frac{\partial^{2+a+b}u_i}{\partial x^{a+1}\partial y^{b+1}}\right)^n_{3\times}\frac{\left(\Delta x_{1+}\right)^a\left(\Delta y_{1+}\right)^b}{a!b!} \\
\left[(u_{i,xy})^n_{5+}\right]^{n-1/2}_{5\times} - (u_{i,xy})^n_{3\times} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\left(\frac{\partial^{2+a+b}u_i}{\partial x^{a+1}\partial y^{b+1}}\right)^n_{3\times}\frac{\left(\Delta x_{5+}\right)^a\left(\Delta y_{5+}\right)^b}{a!b!}
\end{bmatrix}
$$
$$
\text{for } a,b \neq \left\{(0,0),(1,0),(0,1)\right\}. \tag{3.52}
$$

This procedure is then applied to the other two pairs $(1,7)$ and $(5,7)$. After the three possible solutions are generated they are weighted to achieve the values for $u_{i,xyx}$ and $u_{i,xyy}$. There are still values on the RHS of Eq. (3.52) from the current time step but as long as the primary variables are calculated in the correct order these terms would have been calculated. For example consider a fourth-order scheme. In this context the summation terms in Eq. (3.52) disappears leaving the RHS with only the first term $\left[(u_{i,xy})^n_{j+}\right]^{n-1/2}_{j\times} - (u_{i,xy})^n_{3\times}$. The same procedure is also applied to the other third derivatives: $u_{i,xxx}$, $u_{i,xxy}$ $u_{i,yxx}$, $u_{i,yxy}$ $u_{i,yyx}$, and $u_{i,yyy}$. This procedure is easily expanded to any desired order.

3.4.2 Two-Dimensional EBD scheme

The EBD scheme is a recent addition to the CESE toolbox and has been found to be useful for Navier-Stokes simulations where cells may have very large aspect ratios[44, 63]. This scheme uses the same core methodology as the $c - \tau$ scheme but differs in how the expansion points are located.

To begin Fig. (3.6) shows the geometry used in the EBD algorithm.

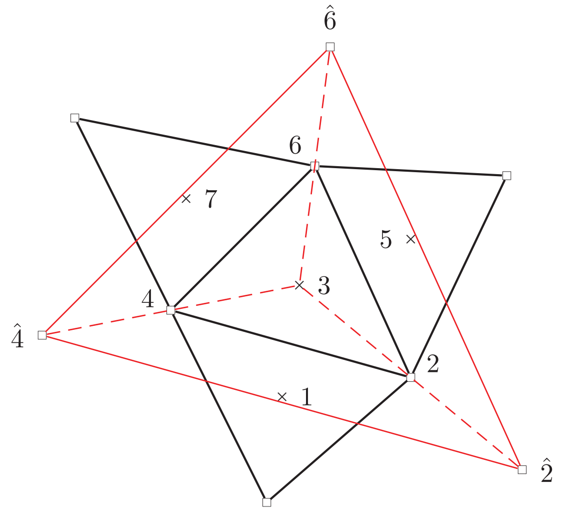


Figure 3.6: The stencils used for the odd derivative EBD scheme.

For ease of notation let $N_0 \equiv 3$, $N_1 \equiv 2$, $N_2 \equiv 6$, $N_3 \equiv 4$ and $\hat{N}_1 \equiv \hat{2}$, $\hat{N}_2 \equiv \hat{6}$, $\hat{N}_3 \equiv \hat{4}$. Then

$$\hat{N}_i = N_0 + \Omega \left(N_i - N_0 \right), \tag{3.53}$$

where Ω is an adjustable parameter that controls the amount of dissipation added. Chang[63] presented an elegant function that takes into account the local quality of the mesh, the CFL number and the presence of a shock when determining an appropriate value of Ω. This sort of function was found to be necessary for meshes with very high aspect ration cells encountered while solving the Navier-Stokes equation near a boundary. But for the purpose of this investigation a simpler function that scales the value of Ω as a function of the local CFL number was found to sufficient for the Euler solver on a relatively uniform mesh. The function used in the majority of the simulations takes the form:

$$\Omega \equiv Max\left(\Omega_{min}, CFL\Omega_{max}\right)$$

Similar to the $c-\tau$ algorithm several possible values of u_x and u_y are calculated based on the pairing of solution and expansion points. For the EBD algorithm the pairing of these points is not as well defined as they where for the $c-\tau$ algorithm. This is the result of the expansion point location being dependent on the vertices and the solution point of the central cell rather than the solution point of the central cell and the solution point of the neighboring cells. For this investigation the solution point expansion point pairing is made up by the solution point and the vertices of its enclosing cell. For a triangular mesh the pairs are (i) $1_\times$ with $\hat{2}$ and $\hat{4}$, (ii) $5_\times$ with $\hat{2}$ and $\hat{6}$, and (iii) $7_\times$ with $\hat{6}$ and $\hat{4}$ The system of equation making up the first system are:

$$\begin{bmatrix} \Delta x_{\hat{2}} & \Delta y_{\hat{2}} \\ \Delta x_{\hat{4}} & \Delta y_{\hat{4}} \end{bmatrix} \begin{bmatrix} (u_{i,x})^n_{3\times} \\ (u_{i,y})^n_{3\times} \end{bmatrix}^{(\hat{2},\hat{4})} = \begin{bmatrix} \left[(u_i)^n_{\hat{2}}\right]^{n-1/2}_{1\times} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\left(\frac{\partial^{a+b}u_i}{\partial x^a \partial y^b}\right)^n_{3\times} \frac{\left(\Delta x_{\hat{2}}\right)^a \left(\Delta y_{\hat{2}}\right)^b}{a!b!} \\ \left[(u_i)^n_{\hat{4}}\right]^{n-1/2}_{1\times} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\left(\frac{\partial^{a+b}u_i}{\partial x^a \partial y^b}\right)^n_{3\times} \frac{\left(\Delta x_{\hat{4}}\right)^a \left(\Delta y_{\hat{4}}\right)^b}{a!b!} \end{bmatrix}, \tag{3.54}$$

where $(a,b) \neq \{(1,0),(0,1)\}$. Applying this to each set yields three possible solutions which are then weighted using a re-weighting algorithm.

The expansion of this method for finding higher odd derivatives follows the same procedure outline for the $c-\tau$ scheme. For example, to determine the values of $u_{i,xxx}$ and $u_{i,xxy}$ Eq. (3.54) takes the form

$$\begin{bmatrix} \Delta x_{\hat{2}} & \Delta y_{\hat{2}} \\ \Delta x_{\hat{4}} & \Delta y_{\hat{4}} \end{bmatrix} \begin{bmatrix} (u_{i,xxx})^n_{3\times} \\ (u_{i,xxy})^n_{3\times} \end{bmatrix}^{(\hat{2},\hat{4})} = \begin{bmatrix} \left[(u_{i,xx})^n_{\hat{2}}\right]^{n-1/2}_{1\times} - (u_{i,xx})^n_{3\times} \\ \left[(u_{i,xx})^n_{\hat{4}}\right]^{n-1/2}_{1\times} - (u_{i,xx})^n_{3\times} \end{bmatrix} \tag{3.55}$$

3.5 Fourth-Order Outline

In this section, the overall time-marching algorithm is illustrated. The computational procedure is organized in the following steps:

1. Calculate all spatial derivatives of $f_i^{x,y}$ in the previous time level based on the

known values of all spatial derivatives of u_i. Calculate all derivatives of u_i and $f_i^{x,y}$ which involve temporal differentiation in the previous time level.

2. Apply the original second-order CESE method four times to calculate the following unknowns: $(u_{i,xx})_j^n$, $(u_{i,xy})_j^n$, $(u_{i,yx})_j^n$, and $(u_{i,yy})_j^n$ at all the solution points in the new time level, e.g., point $3^{\times}$ in Fig. (3.2).

3. Calculate all third derivatives listed in Table 3.1 at all solution points using the finite difference procedure from original second-order CESE method.

4. Calculate the values of $(u_i)_j^n$ at all solution points at the new time step. This is accomplished by integrating the third-order Taylor series expansion of u_i over the CCE.

5. Calculate the first derivatives of $(u_i)_j^n$ at the new time step at all solution points. This is accomplished by applying the central difference procedure to $(u_i)_j^n$.

3.6 Results and Discussions

To assess the accuracy of the newly developed unstructured-mesh solver based on the fourth-order CESE method, the following four benchmark problems were considered: (i) a moving isentropic vortex, (ii) the interaction of an acoustic wave with a entropy wave, (iii) a supersonic flow passing a circular blunt body, and (iv) a supersonic flow over a guttered wedge. Case (i) shows the order of convergence of the numerical results, and cases (ii)-(iv) shows the shock capturing capability of the solver. For all of these simulations the EBD scheme from [44] along with the reweighing scheme Eq. (3.49) where deployed.

3.6.1 Convergence Result

This section reports the numerical result of a moving isentropic vortex. The convergence rate is assessed by measuring the L_1 and L_2 norms of the errors of the calculated u_i. The L_1 and L_2 norms are defined as

$$L_n = \sqrt[n]{\frac{\sum_j \left|(u_i)_{j,\text{num}} - (u_i)_{j,\text{ana}}\right|^n A_j}{\sum_j A_j}} \quad n = 1, 2$$

where the subscript "num" denotes the numerical result and "ana" the analytical solution. In the equation, A_j is the spatial area of cell j.

In the fourth-order CESE method, the third-order Taylor series are used to discretize primary unknowns u_i and the fluxes $f_i^{x,y}$ with $i = 1, \ldots, 4$. Therefore, a fourth-order convergence rate is expected for this case. This simulation is repeated using four different mesh with increasing resolution.

The Euler equations are non-dimensionalized and the initial conditions are taken from Balsara[73]. The initial conditions are:

$$\begin{aligned} u &= u_\infty - \frac{\epsilon}{2\pi} e^{\frac{1}{2}(1-r^2)}\, \bar{y}, \\ v &= v_\infty + \frac{\epsilon}{2\pi} e^{\frac{1}{2}(1-r^2)}\, \bar{x}, \\ T &= T_\infty - \frac{(\gamma-1)\epsilon^2}{8\gamma\pi^2} e^{1-r^2}, \end{aligned} \tag{3.56}$$

where $\bar{y} = y - y_o$, $\bar{x} = x - x_o$, and $r^2 = \bar{x}^2 + \bar{y}^2$. The vortex strength, ϵ, is equal to 5. The center of the vortex is located at (x_o, y_o). The subscript ∞ denote the free-stream conditions. In the present calculation, $T_\infty = 1$ and $u_\infty = v_\infty = 1$. Density, ρ, and pressure, p, are calculated from the isentropic relations and the ideal gas law: $\rho = T^{1/(\gamma-1)}$ and $p = \rho T$. The computational domain is a 10×10 square, and time duration of the simulation is $0 \leq t \leq 10$. The initial conditions of spatial derivatives of u_i are calculated analytically from the initial conditions. Periodic boundary conditions are applied at all boundaries. It should be noted that this problem is not periodic in nature so applying these boundary conditions introduces some numerical error to the simulation.

To determine the convergence rates, the L_1 and L_2 norms of the calculated density were used. The characteristic mesh size, h, is defined as the square root of the averaged cell area. The calculated convergence rates are tabulated in Tables 3.2. As a reference, the convergence rates of numerical solutions obtained by using the original second-order CESE method are tabulated Table 3.3.

Table 3.2: Convergence rates of the isentropic vortex by using the fourth-order CESE method on unstructured meshes.

		CFL			Error		Convergence	
h	dt	min	avg	max	L1	L2	L1	L2
3.28E-01	5.10E-02	0.37	0.47	0.70	1.087E-02	2.674E-02	-	-
1.64E-01	2.55E-02	0.32	0.47	0.73	4.411E-03	1.052E-02	1.30	1.35
1.10E-01	1.70E-02	0.27	0.46	0.70	1.956E-03	6.040E-03	2.01	1.37
8.23E-02	1.28E-02	0.29	0.46	0.71	5.319E-04	1.228E-03	4.52	5.52
6.59E-02	1.02E-02	0.26	0.46	0.71	2.735E-04	7.618E-04	3.00	2.15
5.49E-02	8.50E-03	0.29	0.46	0.70	1.184E-04	3.295E-04	4.57	4.57
4.71E-02	7.29E-03	0.29	0.46	0.70	5.363E-05	1.312E-04	5.15	5.99
4.12E-02	6.38E-03	0.28	0.46	0.72	2.826E-05	6.364E-05	4.83	5.46
3.66E-02	5.67E-03	0.28	0.46	0.70	1.780E-05	3.599E-05	3.93	4.85
3.30E-02	5.10E-03	0.27	0.46	0.74	1.252E-05	2.341E-05	3.36	4.12
3.00E-02	4.64E-03	0.25	0.46	0.76	8.327E-06	1.505E-05	4.29	4.65
2.75E-02	4.25E-03	0.27	0.46	0.73	6.564E-06	1.086E-05	2.75	3.76

The varying convergence rates are not uncommon for unstructured solvers and is quite common in schemes such as the Discontinuous Galerkin. For example the same simulation was solved by the DG scheme by Dumbser and Munz [37] and convergence rates of 4.3 to 4.9 were reported. Balsara et al. [73] reported an even wider range of convergence rates ranging from 2.4 to 6.92 with the use of a limiter and a range of 4.0 to 5.47 without the use of a limiter.

Table 3.3: Convergence rates of the calculated vortex by using the second-order CESE method and unstructured-mesh.

		CFL			Error		Convergence	
h	dt	min	avg	max	L1	L2	L1	L2
3.28E-01	5.10E-02	0.40	0.47	0.63	1.620E-02	4.210E-02	-	-
1.64E-01	2.55E-02	0.34	0.47	0.70	5.899E-03	1.659E-02	1.46	1.35
1.10E-01	1.70E-02	0.29	0.46	0.69	2.921E-03	8.395E-03	1.74	1.69
8.23E-02	1.28E-02	0.30	0.46	0.70	1.786E-03	5.176E-03	1.71	1.68
6.59E-02	1.02E-02	0.27	0.46	0.71	1.186E-03	3.428E-03	1.85	1.86
5.49E-02	8.50E-03	0.29	0.46	0.70	8.727E-04	2.422E-03	1.67	1.89
4.71E-02	7.29E-03	0.29	0.46	0.70	6.835E-04	1.860E-03	1.59	1.72
4.12E-02	6.38E-03	0.28	0.46	0.72	5.489E-04	1.464E-03	1.65	1.80
3.66E-02	5.67E-03	0.28	0.46	0.70	4.610E-04	1.179E-03	1.49	1.84
3.30E-02	5.10E-03	0.27	0.46	0.74	3.848E-04	9.749E-04	1.73	1.82
3.00E-02	4.64E-03	0.25	0.46	0.76	3.259E-04	8.129E-04	1.75	1.91
2.75E-02	4.25E-03	0.27	0.46	0.73	2.903E-04	7.114E-04	1.34	1.54

The L_2 norm for both the second and fourth order errors are shown in Fig. (3.7). In this figure the blue and red lines represent the ideal second- and fourth-order convergence rates respectively.

3.6.2 Acoustic wave interaction

This problem has been used to benchmark problems in Computational Aeroacoustic (CAA)[74]. This problem was original developed to be solved by the linearized Euler equations and was modified slightly for use by the Euler equations. The initial

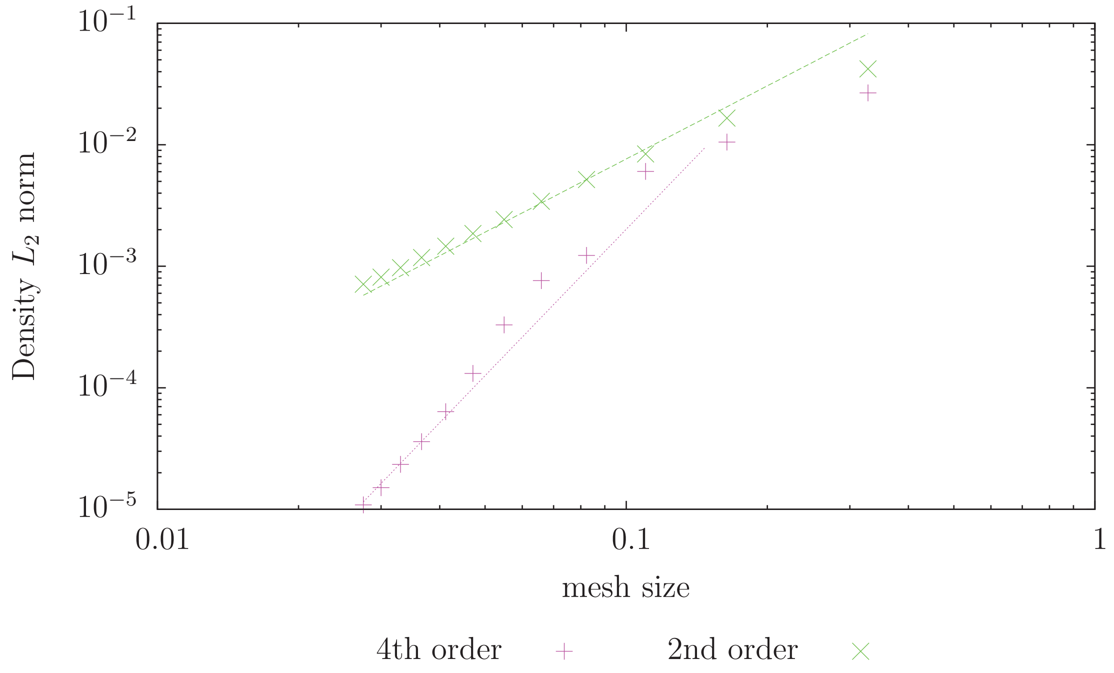


Figure 3.7: The L_2 norms of the calculated density of the simulated vortex based on the fourth- and second-order CESE methods. Where the mesh size is taken as the square root of the average cell area.

conditions used in the simulation are

$$
\begin{aligned}
p'(x,y,t=0) &= \exp\left[-\ln(2)\left(\frac{x^2+y^2}{9}\right)\right] \\
\rho'(x,y,t=0) &= \exp\left[-\ln(2)\left(\frac{x^2+y^2}{9}\right)\right] + 0.1\exp\left[-\ln(2)\left(\frac{(x-67)^2+y^2}{25}\right)\right] \\
u'(x,y,t=0) &= 0.04y\exp\left[-\ln(2)\left(\frac{(x-67)^2+y^2}{25}\right)\right] \\
v'(x,y,t=0) &= -0.04(x-67)\exp\left[-\ln(2)\left(\frac{(x-67)^2+y^2}{25}\right)\right]
\end{aligned}
\tag{3.57}
$$

$$
\begin{aligned}
p = p_\infty + \delta p' \quad \rho = \rho_\infty + \delta\rho' \\
u = u_\infty + \delta u' \quad v = v_\infty + \delta v'
\end{aligned}
\tag{3.58}
$$

where p, ρ, u, v are pressure, density, x and y component of velocity respectively, subscript ∞ denotes the free stream values and a superscript $'$ denotes the perturbation to the free stream and δ is used to decrease the perturbation. In the original investigation $\delta = 1$ and all free stream quantities where zero. In the current simulation $\delta = 10^{-5}$, $p_\infty = 1/\gamma$, $\rho_\infty = 1.0$, $u_\infty = 0.5$, $v = 0$ and $\gamma = 1.4$. The domain is a square with boundaries at ± 100 and non reflecting boundary conditions are applied at all boundaries. The simulation was run for a non-dimensional time of 60. Under these conditions a solution to the linear form of the equation is possible and was originally given by Tam and Webb[75]

$$\begin{aligned}
u'(x,y,t) &= \frac{x-u_\infty t}{2\alpha_1\eta}\int_0^\infty e^{\frac{-\xi^2}{4\alpha 1}}\sin(\xi t)J_1(\xi\eta)\xi d\xi + 0.04ye^{-\alpha_2[(x-67-U_\infty t)^2+y^2]} \\
v'(x,y,t) &= \frac{y}{2\alpha_1\eta}\int_0^\infty e^{\frac{-\xi^2}{4\alpha 1}}\sin(\xi t)J_1(\xi\eta)\xi d\xi - 0.04(x-u_\infty t)e^{-\alpha_2[(x-67-U_\infty t)^2+y^2]} \\
p'(x,y,t) &= \frac{1}{2\alpha_1\eta}\int_0^\infty e^{\frac{-\xi^2}{4\alpha 1}}\cos(\xi t)J_0(\xi\eta)\xi d\xi \\
\rho'(x,y,t) &= p' + 0.1e^{-\alpha_2[(x-67-U_\infty t)^2+y^2]}
\end{aligned} \tag{3.59}$$

Due to the properties of linear equations this solution is still applicable to the modified initial conditions. The simulation was run at three different resolutions, 18K, 74K and 296K cells. The progression of these meshes represents a doubling of resolution. The simulations was run using both the second and fourth-order Euler solver. To determine the accuracy of solver on each mesh the values along the line $y = 0$ is shown in Fig. (3.8). In this figure the results from the second- and fourth-order schemes are compared to the linearized solution. In this plot the domain has been reduced to show greater detail near the pulse interactions. From the results it is evident that for all cases the fourth-order results out perform the second-order results on the same mesh. It can also be seen that the fourth-order results have similar quality as the second-order mesh with twice the resolution e.g. the quality of the results from the fourth-order solver on the 18K mesh are similar to the second-order results from the 74K mesh.

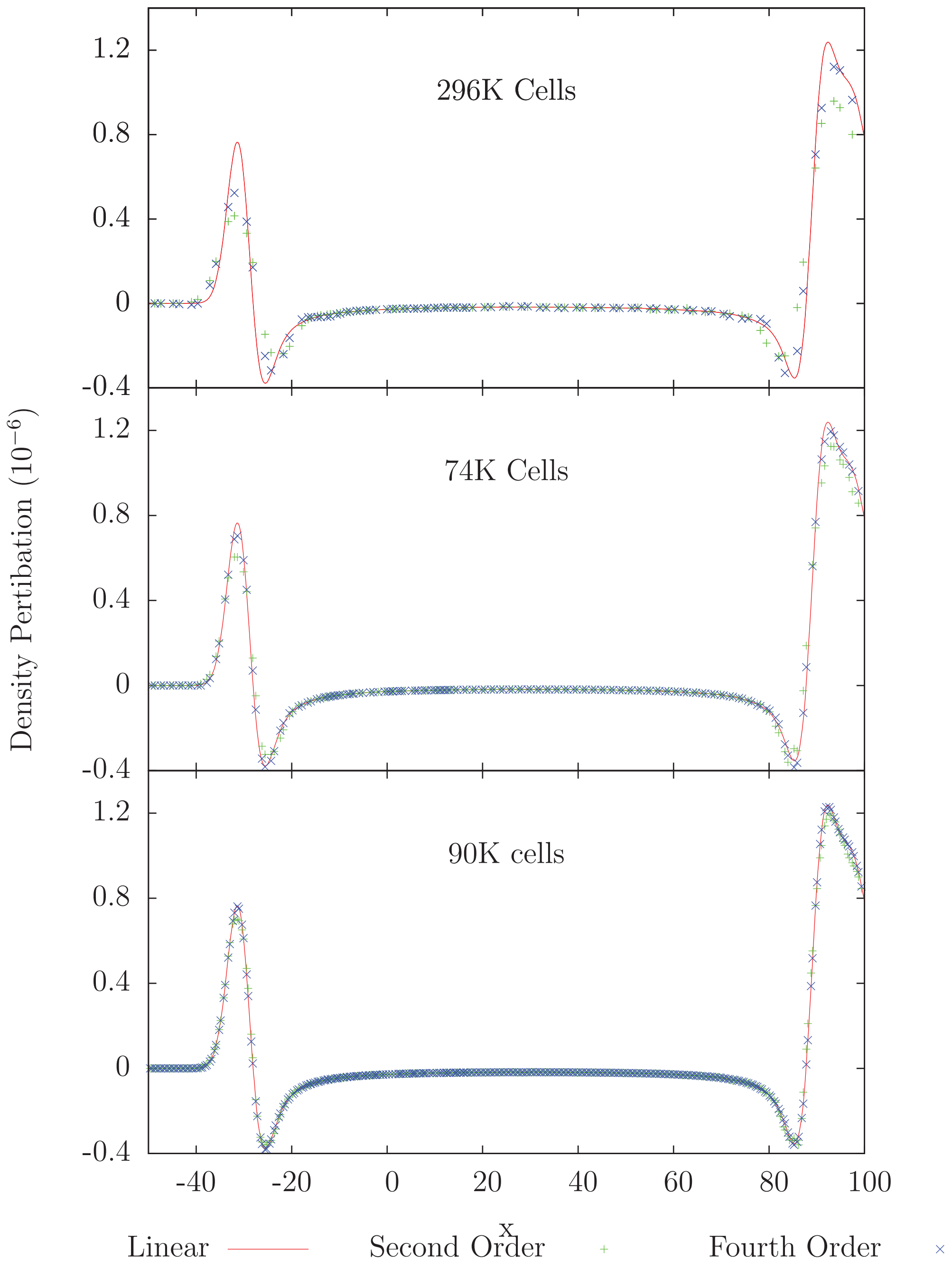


Figure 3.8: Entropy acoustic pulse interaction at t=60.

3.6.3 Supersonic Flow over a Blunt Body

In this subsection, a supersonic flow passing a blunt body is considered. The working fluid is air with a specific heat ratio $\gamma = 1.4$ and the gas constant $R = 287.15$ J/kg K. The free-stream Mach number is 3.0, $p_\infty = 1$ bar, and $\rho_\infty = 1.23$ kg/m^3. The radius of the circular blunt body is 0.5 meters and the outer domain is an ellipse with a major and minor axes of 2.5 and 1.5 meters respectively. The re-weighting scheme employed in this simulation is the S2 scheme from Zhang [72] combined with the geometry from Chang [44].

Figure (3.9) shows the contours of the calculated density of the Mach 3 airflow passing the blunt body. The solid line in the plot indicates the sonic line. The simulation was run with both the second and fourth order schemes at two different resolutions. The coarse mesh has 6148 cells and the refined mesh has 47858 cells. To determine the accuracy the post shock density and the density at the wall were compared to the analytical solution. In all cases the fourth and second order schemes converged to similar solutions for a given mesh. For all cases the post shock density predicted by the simulation was in good agreement with the analytical solution. The density at the stagnation point on the surface predicted by the simulations where 5.07 and 5.24 kg/m^3 for the coarse and fine mesh respectively. These values agree well with the analytical value of 5.29 kg/m^3.

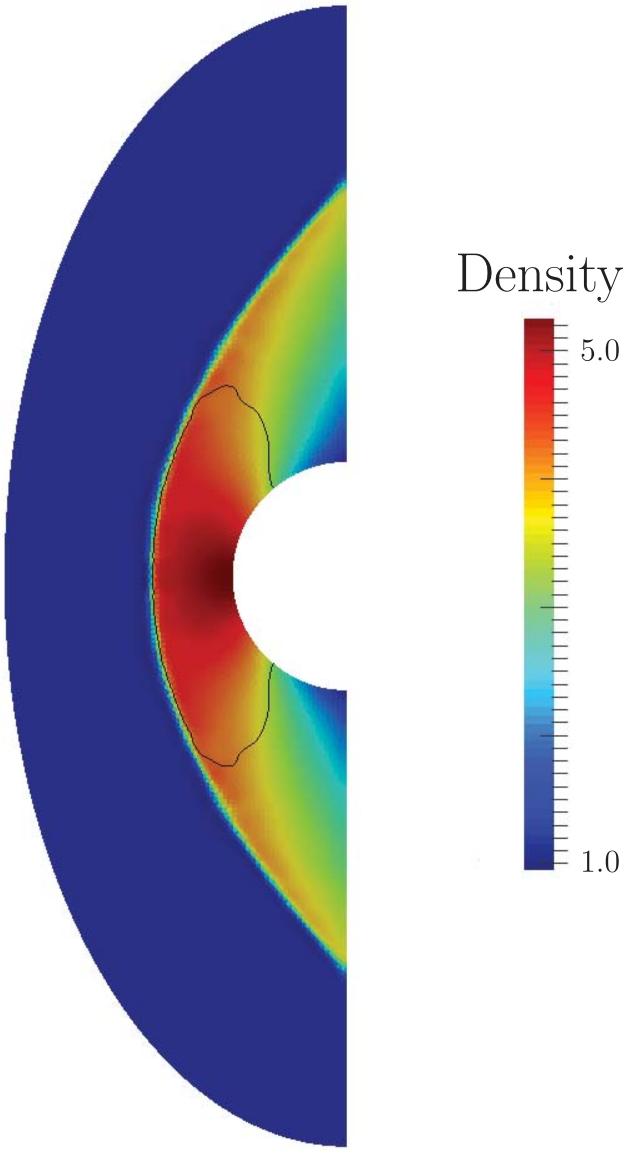


Figure 3.9: The contours of calculated density of a Mach 3 air flow passing a circular blunt body. The black solid line shows the sonic line.

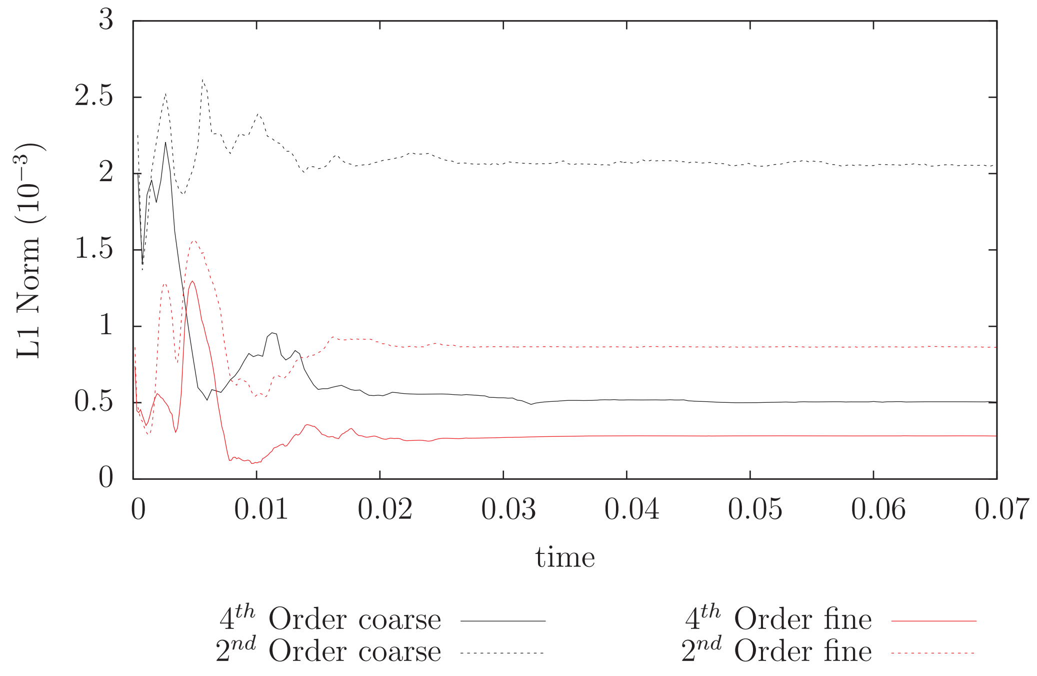
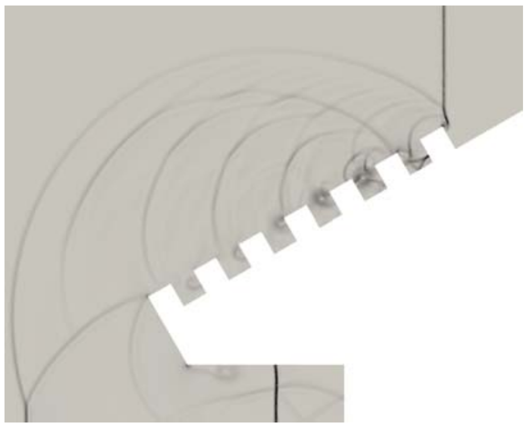
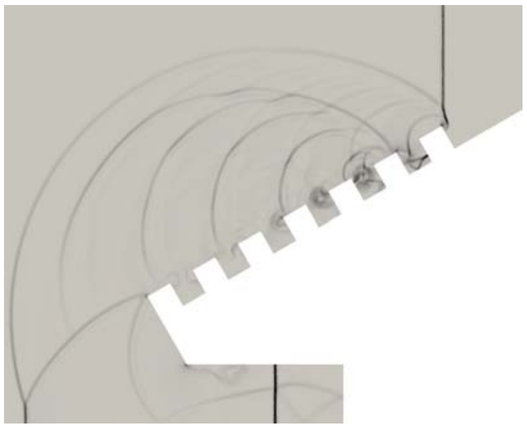


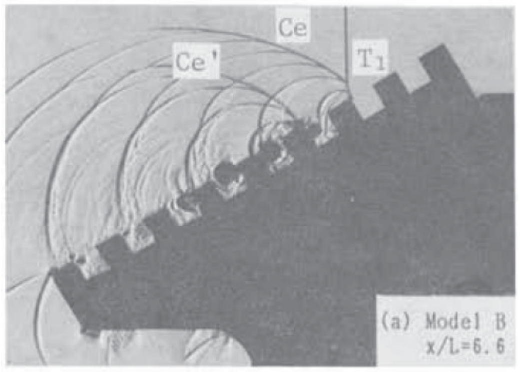
Figure 3.10: The contours of calculated density of a Mach 3 air flow passing a circular blunt body. The black solid line shows the sonic line.

3.6.4 Supersonic Flow over a Guttered Wedge

The final two-dimensional simulation is the so called dust layer problem witch is based on the experiments by Suzuki, et al. [76]. This is an unsteady problem in which supersonic air flows over a wedge with gutters on them. Instead of the standard moving shock wave typical of supersonic flow over a wedge there are many different shock waves that interact with one another as the shock wave progresses up the ramp. This simulation using the same initial conditions and setup as the simulation seen in Suzuki, et al. Fig. (5a)[76]. Under these conditions the wall makes an angle of 30 degrees and each gutter surfaces has a length of 4 mm. The flow ahead of the shock wave had a pressure and temperature of 1 bar and 300 Kelvin respectively. The shock wave moves at a speed of Mach 1.41. The mesh is roughly uniform and contains approximately 144 thousand cells, with this resolution there are approximately 16 cells per gutter surface. The simulation time was 92 micro seconds, which is the time it takes the shock to travel from the start of the wedge to the end of the seventh gutter. A wall boundary condition was applied at all boundaries except the inlet which was subsonic and the exit witch was set to non-reflecting. During the simulation the CFL number ranged from about 0.25 to 1.1. Since the calculation of the CFL number is not exact having a CFL number slightly greater than 1 is not unexpected. To compare with the experimental results a numerical Schlieren was computed by $\left(\sqrt{\left(\frac{\partial\rho}{\partial x}\right)^2+\left(\frac{\partial\rho}{\partial y}\right)^2}\right)$.


(a) Second Order


(b) Fourth Order



(c) Experimental Schlieren [76]

Figure 3.11: Numerical Schlieren contours for a Mach 1.41 flow over a guttered wall.

CHAPTER 4

THREE-DIMENSIONAL CESE

To begin, in the three-dimensional derivation it is assumed that a set of N_{eq} three-dimensional, coupled, first-order hyperbolic partial differential equations are cast into the following vector form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^x}{\partial x} + \frac{\partial \mathbf{F}^y}{\partial y} + \frac{\partial \mathbf{F}^z}{\partial z} = 0 \tag{4.1}$$

where $\mathbf{U} = (u_1, u_2, \ldots, u_{N_{eq}})^t$ is the unknown vector, and $\mathbf{F}^{x,y,z} = (f_1^{x,y,z}, f_2^{x,y,z}, \ldots, f_{N_{eq}}^{x,y,z})^t$, is the flux function vector. In contrast to conventional finite volume methods, the Reynold-Leibniz integration rule, also known as the transport theorem, is not used to form the equivalent integral equations for conservation laws. Instead, in the CESE method, each equation in Eq. (4.1) is perceived as a divergence-free condition in the four-dimensional Euclidean space,$\mathcal{E}_4$, (x, y, z, t):

$$\nabla \cdot \mathbf{h}_i = 0, \tag{4.2}$$

where $\mathbf{h}_i = (f_i^x, f_i^y, f_i^z, u_i)^t$ is the space-time flux function and $i = 1, \ldots, N_{eq}$. Aided by Gauss' theorem, Eq. (4.1) is equivalent to the following integral form:

$$\oint \mathbf{h}_i(x, y, z, t) \cdot d\mathbf{s} = 0. \tag{4.3}$$

In the CESE method, Eq. (4.3) is integrated over each CE with $\mathbf{s}$ as the surface of the CE. As such, space-time flux conservation over each CE is enforced.

4.1 Three-Dimensional Discretization

The previous two chapters began with a detailed schematics of the CE and SE. Unfortunately, the CE and SE associated with the three-dimensional CESE scheme are too complex to clearly illustrate. Instead, the CE is broken up into three parts: (i) the top BCE, (ii) the bottom BCE, and (iii) the side BCE. For readability, the following derivation will assume that the mesh only contains tetrahedrals. Once this derivation is understood it is trivial to extend to meshes with other cell types including mixed elements.

To begin, consider a tetrahedral mesh where each cell has four vertices and one centroid, C_i. For easy of notation each tetrahedral is referred to by its centroid, e.g. tetrahedral 3 has as centroid C_3. The construction of the three-dimensional CE follows the same basic steps of its two-dimensional counterpart and consists of two primary steps. First, the bottom BCE is constructed and is then projected in time by a distance of $\Delta t/2$. Since this derivation assumes a non-moving mesh the bottom and top BCEs have the same spatial dimensions but different temporal ones.

To construct the bottom BCE, consider a system of five tetrahedrals C_i, $i = 0, \ldots, 4$. Where C_0 is the central tetrahedral and C_1, C_2, C_3, and C_4 are its neighbors. At this point the bottom BCE is further divided into four parts, where each part is associated with one of the neighboring cells. It is the union of these four parts that will form the bottom BCE. The shape of each of the four parts is a triangular bipyramid and its vertices consists of: the cell centroid of the central cell, the cell centroid of a neighboring cell and the three vertices that makes up the face shared by the central and neighboring cells. Although this shape has the same number of vertices as a four-sided pyramid, no four points coexist on the same plane. For example, consider the points C_0 and C_3, as well as the three vertices that makes up the face shared by C_0 and C_3, i.e. V_1, V_2, and V_4. This portion of the BCE is shown in Fig. (4.1). The other three parts of the bottom BCE are formed by taking C_0 a neighboring cell center, C_1, C_2, and C_4 and the vertices that makes up their shared face.

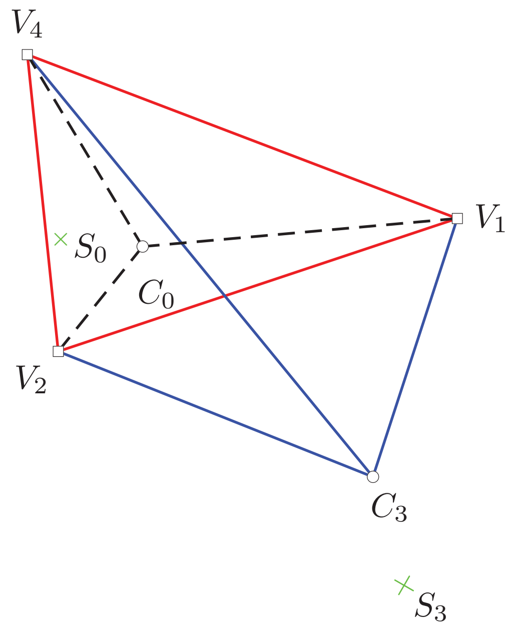


Figure 4.1: A portion of the bottom BCE for a three-dimensional tetrahedral mesh.

Two important points in this figure are the solution points associated with cells C_0 and C_3. These points are denoted by a green $\times$ and the labeled as S_0 and S_3. The location of the solution points are found using the same procedure used in the second-order CESE scheme. The use of various line colors is meant to aid the reader in identifying the different segments. The red lines are the shared faces of cells C_0 and C_3, the blue lines are formed by the vertices and neighboring cell center and the dashed black lines are formed by the central cells' cell center and the vertices of the shared face.

The next step is to construct the side BCEs. To do so, take one of the faces from one of the bottom parts of the BCE and extrude it in time forming a BCE in a four-dimensional space-time coordinate system. A sample side BCE is shown in Fig. (4.2). In this figure the points with an apostrophe, $'$, denote the position at the next time step. It is important to note that the spatial coordinates do not change, e.g. $x_{V_1} = x'_{V_1}$, $y_{V_1} = y'_{V_1}$, $z_{V_1} = z'_{V_1}$.

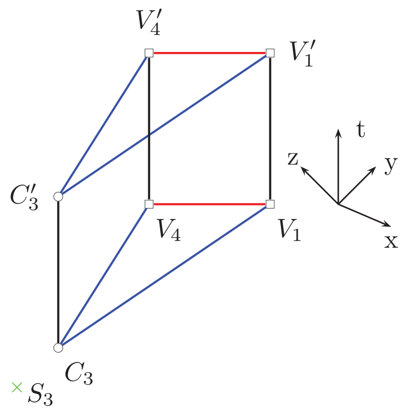


Figure 4.2: A side BCE associated with the solution point C_3.

It should be noted that there are six side BCEs associated with each portion of the bottom BCE. Making a grand total of twenty four side BCEs.

The final step is the formation of the top BCE. This is formed by the union of all vertices from all of the side BCEs at the next time step while maintaining the connectivity of the vertices. The resulting object is identical to the bottom BCE except that the temporal coordinates are shifted by $\Delta t/2$. A portion of the top BCE is shown in Fig. (4.3).

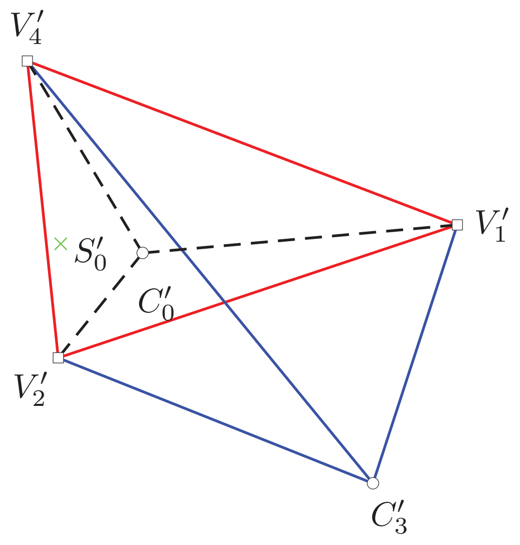


Figure 4.3: A top BCE segment of a for a three-dimensional tetrahedral mesh BCE.

The CCE associated with cell C_0 is formed by the union of all bottom, side and top BCEs described above. This CCE can be divided into four BCEs, with each one associated with one of the neighboring cells. For example the bottom and top portions of the BCE associated with C_3 are show in in Fig. (4.1) and Fig. (4.3) respectively and the side portions of the BCE consist of the six side BCEs detailed above.

With the CE defined in three-dimensions it is necessary to define the SE. In the three-dimensional CESE scheme the SE is a four-dimensional volume and cannot be properly represented on paper. A formal definition of a SE for the three-dimensional CESE scheme is also difficult to express. Instead it is more productive to think of which portions of the CCE are governed by which solution point. To begin, the Taylor series that is expanded from the neighboring solution points are used to integrate over the bottom BCE as well as the outer portions of the side BCE. The out portions of the side BCE are those that does not contain the points C_0 and C_0'. The Taylor series expanded from the central solution point at the new time step is used to integrate over the top BCE as well as the interior side BCEs. Where the interior side BCEs are those that contain the points C_0 and C_0'.

With the CE and SE defined it is necessary to express the discretization of the primary variables. To accomplish this an Nth-order Taylor series expansion of the unknown u_i inside a SE is used. This Taylor series is expressed as:

$$u_i^* = \sum_{a=0}^{N} \sum_{b=0}^{N-a} \sum_{c=0}^{\substack{N-\\a-b}} \sum_{d=0}^{\substack{N-a-\\b-c}} \left(\frac{\partial^{a+b+c+d} u_i}{\partial x^a \partial y^b \partial z^c \partial t^d} \right)_j^n \frac{(x-x_j)^a}{a!} \frac{(y-y_j)^b}{b!} \frac{(z-z_j)^c}{c!} \frac{(t-t^n)^d}{d!} \tag{4.4}$$

The subscript j and the superscript n are the spatial and temporal anchoring point respectively. Since the current investigation does not assume the symmetry property of spatial derivatives, variables such as $\frac{\partial^2 u}{\partial x \partial y}$ and $\frac{\partial^2 u}{\partial y \partial x}$ are considered two distinct variables. This means that Eq. (4.4) is not initially compatible with this assumption.

In order to use Eq. (4.4) within the new scheme the mixed derivatives are averaged. For example

$$\frac{\partial^3 u_i}{\partial x^2 \partial z} \equiv \frac{1}{3}\left(\frac{\partial^3 u_i}{\partial x \partial x \partial z} + \frac{\partial^3 u_i}{\partial x \partial z \partial x} + \frac{\partial^3 u_i}{\partial z \partial x \partial x}\right).$$

All derivatives of u_i in Eq. (4.4) are succinctly expressed by the following Taylor series expansion:

$$\frac{\partial^C u_i}{\partial x^I \partial y^J \partial z^K \partial t^L} = \sum_{a=0}^{A} \sum_{b=0}^{A-a} \sum_{c=0}^{\substack{A- \\ a-b}} \sum_{d=0}^{\substack{A- \\ a-b-c}} \left(\frac{\partial^B u_i}{\partial x^{I+a} \partial y^{J+b} \partial z^{K+c} \partial t^{L+d}}\right)_j^n \frac{(x-x_j)^a}{a!} \frac{(y-y_j)^b}{b!} \frac{(z-z_j)^c}{c!} \frac{(t-t^n)^d}{d!} \tag{4.5}$$

where $C = I + J + K + L, A = N - C$, and $B = C + a + b + c + d$. Obviously, the Taylor series expansion of u_i in Eq. (4.4) is a special case of Eq. (4.5) with $A = N$ and $C = 0$. Similarly, the fluxes, $f_i^{x,y,z}$, and their derivatives inside a SE are also discretized by the Taylor series expansion:

$$\frac{\partial^C f_i^{x,y,z}}{\partial x^I \partial y^J \partial z^K \partial t^L} = \sum_{a=0}^{A} \sum_{b=0}^{A-a} \sum_{c=0}^{\substack{A- \\ a-b}} \sum_{d=0}^{\substack{A- \\ a-b-c}} \left(\frac{\partial^B f_i^{x,y,z}}{\partial x^{I+a} \partial y^{J+b} \partial z^{K+c} \partial t^{L+d}}\right)_j^n \frac{(x-x_j)^a}{a!} \frac{(y-y_j)^b}{b!} \frac{(z-z_j)^c}{c!} \frac{(t-t^n)^d}{d!} \tag{4.6}$$

where C and B have the same definitions as that in Eq. (4.5). When $C = 0$, Eq. (4.6) is the Taylor series expansion of the flux $f_i^{x,y,z}$ itself.

The Taylor series coefficients listed in Eqs. (4.5) and (4.6) are the unknown variables that need to be calculated as part of the Higher Order CESE method.

4.2 Three-Dimensional Independent Variables

In the following derivation it will be shown that the only independent variables are the conserved variables and their spatial derivatives. This portion of the derivation follows the same procedure used in the one- and two-dimensional derivations.

To begin, the flux terms are written as functions of u_i and its spatial and temporal derivatives. As it was with the one- and two-dimensional algorithms two different methods will be presented. The first method, shown below, is generic and not tide to any particular physics while the second method, shown in Section 5.2, is more numerically efficient but its implementation is dependent on the constitutive equation, e.g. Euler, Navier-Stokes.

To begin, the chain rule is used to determine the first derivatives of $f_i^{x,y,z}$ and is expressed by the summation:

$$\frac{\partial f_i^{x,y,z}}{\partial \Psi_1} = \sum_{l=1}^{m} \frac{\partial f_i^{x,y,z}}{\partial u_l} \frac{\partial u_l}{\partial \Psi_1}, \tag{4.7}$$

where $\Psi_1 = \{x, y, z\}$, and t. On the right hand side of Eq. (4.7), $\partial f_i^{x,y,z}/\partial u_l$ is the Jacobian matrix, which is easily derived from the relationship between the fluxes and the conserved quantities. For the second derivatives,

$$\frac{\partial^2 f_i^{x,y,z}}{\partial\Psi_1\partial\Psi_2} = \sum_{l=1}^{m} \frac{\partial f_i^{x,y,z}}{\partial u_l}\frac{\partial^2 u_l}{\partial\Psi_1\partial\Psi_2} + \sum_{l=1}^{m}\sum_{p=1}^{m} \frac{\partial^2 f_i^{x,y,z}}{\partial u_l \partial u_p}\frac{\partial u_l}{\partial\Psi_1}\frac{\partial u_p}{\partial\Psi_2}, \tag{4.8}$$

where $(\Psi_1, \Psi_2) = \{(x,x),(x,y),(x,z),(x,t),(y,x),(y,y),(y,z),(y,t),(z,x),(z,y),$ $(z,z),(z,t),(t,t)\}$. The second term on the right hand side of Eq. (4.8), $\partial^2 f_i^{x,y,z}/\partial u_l \partial u_p$ is a $N_{eq} \times N_{eq} \times N_{eq}$, matrix, which is readily derived from the governing equations. For the third derivatives,

$$\begin{aligned}\frac{\partial^3 f_i^{x,y,z}}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} =& \sum_{l=1}^{m} \frac{\partial f_i^{x,y,z}}{\partial u_l}\frac{\partial^3 u_l}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} + \\ & \sum_{l=1}^{m}\sum_{p=1}^{m} \frac{\partial^2 f_i^{x,y,z}}{\partial u_l\partial u_p}\left(\frac{\partial^2 u_l}{\partial\Psi_1\partial\Psi_2}\frac{\partial u_p}{\partial\Psi_3} + \frac{\partial^2 u_l}{\partial\Psi_1\partial\Psi_3}\frac{\partial u_p}{\partial\Psi_2} + \frac{\partial^2 u_l}{\partial\Psi_2\partial\Psi_3}\frac{\partial u_p}{\partial\Psi_1}\right) + \\ & \sum_{l=1}^{m}\sum_{p=1}^{m}\sum_{q=1}^{m} \frac{\partial^3 f_i^{x,y,z}}{\partial u_l\partial u_p\partial u_q}\frac{\partial u_l}{\partial\Psi_1}\frac{\partial u_p}{\partial\Psi_2}\frac{\partial u_q}{\partial\Psi_3},\end{aligned} \tag{4.9}$$

$$
\begin{aligned}
(\Psi_1, \Psi_2, \Psi_3) = \{&(x,x,x),\ (x,x,y),\ (x,x,z),\ (x,x,t),\\
&(x,y,x),\ (x,y,y),\ (x,y,z),\ (x,y,t),\\
&(x,z,x),\ (x,z,y),\ (x,z,z),\ (x,z,t),\\
&(y,x,x),\ (y,x,y),\ (y,x,z),\ (y,x,t),\\
&(y,y,x),\ (y,y,y),\ (y,y,z),\ (y,y,t),\\
&(y,z,x),\ (y,z,y),\ (y,z,z),\ (y,z,t),\\
&(z,x,x),\ (z,x,y),\ (z,x,z),\ (z,x,t),\\
&(z,y,x),\ (z,y,y),\ (z,y,z),\ (z,y,t),\\
&(z,z,x),\ (z,z,y),\ (z,z,z),\ (z,z,t),\\
&(x,t,t),\ (y,t,t),\ (z,t,t),\ (t,t,t)\}.
\end{aligned}
$$

The last term on the right hand side of Eq. (4.9), $\partial^3 f_i^{x,y,z}/\partial u_l \partial u_p \partial u_q$ is a $N_{eq} \times N_{eq} \times N_{eq} \times N_{eq}$ matrix, which is derived based in the definition of $f_i^{x,y,z}$ as functions of u_i. Aided by Eqs. (4.7-4.9), all of the derivatives of $f_i^{x,y,z}$ are expressed as functions of u_i and its derivatives.

Next, the Cauchy-Kowalewski procedure for non-linear equations is used to relate the temporal derivatives of the primary unknowns to the conserved variable u_i as well as its spatial derivatives e.g. $u_{i,x}$, $u_{i,xy}$, $u_{i,xyz}$,.... To proceed, let the Taylor series satisfy Eq. (4.1) at point (j,n),

$$\left(\frac{\partial u_i}{\partial t}\right)_j^n = -\left(\frac{\partial f_i^x}{\partial x}\right)_j^n - \left(\frac{\partial f_i^y}{\partial y}\right)_j^n - \left(\frac{\partial f_i^z}{\partial z}\right)_j^n \tag{4.10}$$

Each term in the above equation is a coefficient in the Taylor series. For conciseness, the super- and sub-scripts, i.e., j and n, indicating the space-time location are dropped in the following derivation. Aided by Eq. (4.10), the second derivatives of u_i involving the first time differentiation are readily available by assuming that the following algebraic equations are valid:

$$\begin{aligned}\frac{\partial^2 u_i}{\partial x\partial t} &= -\frac{\partial^2 f_i^x}{\partial x\partial x} - \frac{\partial^2 f_i^y}{\partial y\partial x} - \frac{\partial^2 f_i^z}{\partial z\partial x}, \quad & \frac{\partial^2 u_i}{\partial y\partial t} &= -\frac{\partial^2 f_i^x}{\partial x\partial y} - \frac{\partial^2 f_i^y}{\partial y\partial y} - \frac{\partial^2 f_i^z}{\partial z\partial y}, \\ \frac{\partial^2 u_i}{\partial z\partial t} &= -\frac{\partial^2 f_i^x}{\partial x\partial z} - \frac{\partial^2 f_i^y}{\partial y\partial z} - \frac{\partial^2 f_i^z}{\partial z\partial z}, \quad & \frac{\partial^2 u_i}{\partial t\partial t} &= -\frac{\partial^2 f_i^x}{\partial x\partial t} - \frac{\partial^2 f_i^y}{\partial y\partial t} - \frac{\partial^2 f_i^z}{\partial z\partial t},\end{aligned} \tag{4.11}$$

Although Eq. (4.11) has a temporal derivative of u_i on the right hand side of the equation, it is still a function of spatial derivatives of u_i. As an example, the first term on the right hand side of the equation, i.e., $\left(\frac{\partial^2 f_i^x}{\partial x\partial t}\right)$ is examined. This term is easily recast into the following expression:

$$\begin{aligned}\frac{\partial^2 f_i^x}{\partial x\partial t} &= \sum_{l=1}^{m}\frac{\partial f_i^x}{\partial u_l}\frac{\partial^2 u_l}{\partial x\partial t} + \sum_{l=1}^{m}\sum_{p=1}^{m}\frac{\partial^2 f_i^x}{\partial u_l\partial u_p}\frac{\partial u_l}{\partial x}\frac{\partial u_p}{\partial t} \\ &= -\sum_{l=1}^{m}\frac{\partial f_i^x}{\partial u_l}\left(\frac{\partial^2 f_i^x}{\partial x\partial x} + \frac{\partial^2 f_i^y}{\partial y\partial x} + \frac{\partial^2 f_i^z}{\partial z\partial x}\right) - \\ &\qquad \sum_{l=1}^{m}\sum_{p=1}^{m}\frac{\partial^2 f_i^x}{\partial u_l\partial u_p}\frac{\partial u_l}{\partial x}\left(\frac{\partial f_i^x}{\partial x} + \frac{\partial f_i^y}{\partial y} + \frac{\partial f_i^z}{\partial z}\right).\end{aligned}$$

This procedure is then applied to the succeeding terms in Eq. (4.11).

For the third derivatives of u_i involving a single temporal derivatives it is assumed that:

$$\begin{aligned}
\frac{\partial^3 u_i}{\partial x \partial \Psi \partial t} &= -\frac{\partial}{\partial \Psi}\left(\frac{\partial^2 f_i^x}{\partial x \partial x} + \frac{\partial^2 f_i^y}{\partial y \partial x} + \frac{\partial^2 f_i^z}{\partial z \partial x}\right), \\
\frac{\partial^3 u_i}{\partial y \partial \Psi \partial t} &= -\frac{\partial}{\partial \Psi}\left(\frac{\partial^2 f_i^x}{\partial x \partial y} + \frac{\partial^2 f_i^y}{\partial y \partial y} + \frac{\partial^2 f_i^z}{\partial y \partial y}\right), \\
\frac{\partial^3 u_i}{\partial z \partial \Psi \partial t} &= -\frac{\partial}{\partial \Psi}\left(\frac{\partial^2 f_i^x}{\partial x \partial z} + \frac{\partial^2 f_i^y}{\partial y \partial z} + \frac{\partial^2 f_i^z}{\partial z \partial z}\right), \\
\frac{\partial^3 u_i}{\partial t \partial t \partial t} &= -\frac{\partial}{\partial t}\left(\frac{\partial^2 f_i^x}{\partial x \partial t} + \frac{\partial^2 f_i^y}{\partial y \partial t} + \frac{\partial^2 f_i^z}{\partial z \partial t}\right),
\end{aligned} \tag{4.12}$$

where $\Psi = \{x, y, z, t\}$. Equation (4.12) includes terms with multiple temporal derivatives on the right hand side of the equation. These terms are expressed in terms of spatial derivatives through the same approach as that shown in Eq. (4.11).

Essentially, Eqs. (4.10-4.12) assume that the coefficients of Taylor series expansion for u_i and $f_i^{x,y,z}$ satisfy the additional higher-order equations, which are readily obtained by applying spatial differentiation to the Euler equations Eq. (4.1). The procedure is recursive and is extended to higher derivatives of u_i.

To recap, all the first-, second-, and third derivatives of u_i and $f_i^{x,y,z}$ involving any derivative of time can always be replaced by relationship formulated in terms of spatial derivatives of u_i: This is achieved by the following steps: (i) the additional equations shown in Eqs. (4.10-4.12), (ii) Eqs. (4.7-4.9), and (iii) the chain rule as shown in Eqs. (4.7-4.9). As such, the independent variables in the fourth-order CESE method only include the conserved variables u_i with $i = 1, 2, \ldots, N_{eq}$ and their spatial derivatives. Table 4.1 lists the primary unknowns of the fourth-order CESE method.

Table 4.1: The list of unknowns for the three-dimensional, fourth-order CESE method.

Even variables	Odd variables		
u_i	$u_{i,x}$	$u_{i,y}$	$u_{i,z}$
$u_{i,xx}$	$u_{i,xxx}$	$u_{i,xxy}$	$u_{i,xxz}$
$u_{i,xy}$	$u_{i,xyx}$	$u_{i,xyy}$	$u_{i,xyz}$
$u_{i,xz}$	$u_{i,xzx}$	$u_{i,xzy}$	$u_{i,xzz}$
$u_{i,yx}$	$u_{i,yxx}$	$u_{i,yxy}$	$u_{i,yxz}$
$u_{i,yy}$	$u_{i,yyx}$	$u_{i,yyy}$	$u_{i,yyz}$
$u_{i,yz}$	$u_{i,yzx}$	$u_{i,yzy}$	$u_{i,yzz}$
$u_{i,zx}$	$u_{i,zxx}$	$u_{i,zxy}$	$u_{i,zxz}$
$u_{i,zy}$	$u_{i,zyx}$	$u_{i,zyy}$	$u_{i,zyz}$
$u_{i,zz}$	$u_{i,zzx}$	$u_{i,zzy}$	$u_{i,zzz}$

4.3 Three-Dimensional Space-Time Integration

In this section, the procedure for enforcing space-time flux conservation in three-dimensions is illustrated. The integration is used to calculate the primary unknowns and their even derivatives and is carried out in a four-dimensional Euclidean space, three in space and one in time.

The integration is split into three parts: (i) the side BCE, (ii) the bottom BCE, and (iii) top BCE. The flux through the top BCE is a function of the solution at the new time step. On the other hand, the flux through the side and bottom BCEs are calculated from the known solution at the previous time step at neighboring points of the current solution point. Even though the integration over the top BCE uses values at the new time step the integration is still explicit as long as the values at the new time step are calculated in the correct order.

4.3.1 Three-Dimensional Side Integration

The integration over the side BCEs requires three separate integrations for each BCE, one for each external face. For a tetrahedral mesh there are four BCEs resulting in a total of 12 separate integrations. Since the integration over each side BCE is identical the procedure is only shown once. To generalize the derivation let point 1 be the neighboring cell center, points 2 and 3 be the vertices at the previous time step and points $1'$, $2'$, and $3'$ be the same coordinates at the new time step, and $(j_r; t^{n-1/2})$ is the coordinate of the neighboring cell center.

The computation of the flux through one segment of the side BCE requires the evaluation of the integral in Eq. (4.3). The first step in this integration is to determine the normal vector for this BCE. The normal was found to be:

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} & \hat{t} \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 & t_2 - t_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 & t_3 - t_1 \\ x_1' - x_1 & y_1' - y_1 & z_1' - z_1 & t_1' - t_1 \end{vmatrix} \equiv [N^x\ \hat{i}, N^y\ \hat{j}, N^z\ \hat{k}, 0\ \hat{t}] \tag{4.13}$$

It is important to note that the normal in the $\hat{t}$ direction is zero. Applying Eq. (4.13) to Eq. (4.3) yields:

$$
\begin{aligned}
\left[(F_i)^{3D}_{side}\right]_{j_r} = & \sum_{x^i}^{x,y,z} \sum_{a=0}^{N} \sum_{b=0}^{N-a} \sum_{c=0}^{N-a-b} \sum_{d=0}^{N-a-b-c} \left(\frac{\partial f_i^{x^i}}{\partial x^a \partial y^b \partial z^c \partial t^d} \right)^{n-1/2}_{j_r} \frac{1}{a!b!c!d!} \\
& n^{x^i}_{S1} \int_{V^{4D}_{S1}} \left(x - x_{j_r}\right)^a \left(y - y_{j_r}\right)^b \left(z - z_{j_r}\right)^c \left(t - t^{n-1/2}\right)^d dV^4_{S1} + \\
& n^{x^i}_{S2} \int_{V^{4D}_{S2}} \left(x - x_{j_r}\right)^a \left(y - y_{j_r}\right)^b \left(z - z_{j_r}\right)^c \left(t - t^{n-1/2}\right)^d dV^4_{S2} + \\
& n^{x^i}_{S3} \int_{V^{4D}_{S3}} \left(x - x_{j_r}\right)^a \left(y - y_{j_r}\right)^b \left(z - z_{j_r}\right)^c \left(t - t^{n-1/2}\right)^d dV^4_{S3},
\end{aligned}
\tag{4.14}
$$

where $n^{x^i}_{Sj}$ is the normalized unit vector and V^{4D}_{Sj}, $j = 1, 2, 3$ represents each of the side BCEs associated with a single BCE. This integration is repeated for each BCE. For the side BCE associated with solution point S_3 shown in Figures (4.2) and (4.1) each side BCE is composed of:

- $V^{4D}_{S1} \equiv \{C_3, V_1, V_4, C'_3, V'_1, V'_4\}$
- $V^{4D}_{S2} \equiv \{C_3, V_4, V_2, C'_3, V'_4, V'_2\}$
- $V^{4D}_{S3} \equiv \{C_3, V_2, V_1, C'_3, V'_2, V'_1\}$

Equation (4.14) can be simplified by recognizing that the temporal integration is decoupled from the spatial integration. Further simplification is possible by expressing Eq. (4.14) in its parametric form, e.g., $x = x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta$. These simplifications leads to the equation,

$$\left[(F_i)^{3D}_{side} \right]_{j_r} = \sum_{x^i}^{x,y,z} \sum_{a=0}^{N} \sum_{b=0}^{N-a} \sum_{c=0}^{\substack{N- \\ a-b}} \sum_{d=0}^{\substack{N-a- \\ b-c}} \left(\frac{\partial f_i^{x^i}}{\partial x^a \partial y^b \partial z^c \partial t^d} \right)_{j_r}^{n-1/2} \frac{1}{a!b!c!(d+1)!} \left(\frac{\Delta t}{2} \right)^{d+1}$$
$$\sum_{Si}^{1,2,3} N_{Si}^{x^i} \int_0^1 \int_0^{1-\xi} \left[(x_2 - x_1)\,\xi + (x_3 - x_1)\,\eta + x_1 - x_j \right]^a$$
$$\left[(y_2 - y_1)\,\xi + (y_3 - y_1)\,\eta + y_1 - y_j \right]^b \left[(z_2 - z_1)\,\xi + (z_3 - z_1)\,\eta + z_1 - z_j \right]^c d\eta d\xi, \tag{4.15}$$

where $\sum_{Si}^{1,2,3}$ represents the summation over each of the side BCEs associated with a single BCE. Since Eq. (4.15) will be referenced later it is convenient to express it as: [Eq. (4.15)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_j, \Delta t, f_{jcl}^{x^i})$. Equation (4.15) is more complex to integrate than its two-dimensional counter part but is still easily computed using a symbolic mathematical package such as MATLAB. The integral in Eq. (4.15) is integrated below for the values of a, b and c required by the fourth-order CESE algorithm. To start let $\Delta x_i = x_i - x_j$, $\Delta y_i = y_i - y_j$, $\Delta z_i = z_i - z_j$ for $i = 1, 2, 3$. With this definition the integral in Eq. (4.15) becomes

$$\mathcal{L}_{a,b,c}^{s} = \int_0^1 \int_0^{1-\xi} \prod_{j=1}^{3} \left[\Delta\Psi_2^j \xi + \Delta\Psi_3^j \eta + \Delta\Psi_1^j (1 - \xi - v) \right]^{\omega^j} d\eta d\xi,$$

where Ψ is any combination of $\{x, y, z\}$ and $\boldsymbol{\omega} = \{a, b, c\}$. When $a = b = c = 0$ the integration is

$$\mathcal{L}^{s3D}_{0,0,0} = \frac{1}{2}.$$

To integrate any of the first derivatives let $a = 1$ and $b = c = 0$ the integration is

$$\mathcal{L}^{s3D}_{1,0,0} = \frac{1}{6}\sum_{i=1}^{3}\Delta x_i.$$

For any second derivatives let $a = b = 1$ and $c = 0$

$$\mathcal{L}^{s3D}_{1,1,0} = \frac{1}{24}\left[\sum_{i=1}^{3}(\Delta x_i \Delta y_i) + \sum_{i=1}^{3}\Delta x_i \sum_{i=1}^{3}\Delta y_i\right].$$

For any third derivative let $a = b = c = 1$

$$\begin{aligned}\mathcal{L}^{s3D}_{1,1,1} = \frac{1}{120}\Big[&\sum_{i=1}^{3}\Delta x_i \sum_{i=1}^{3}\Delta y_i \sum_{i=1}^{3}\Delta z_i + 2\sum_{i=1}^{3}(\Delta x_i \Delta y_i \Delta z_i) + \\ &\sum_{i=1}^{3}(\Delta x_i \Delta y_i)\sum_{i=1}^{3}\Delta z_i + \sum_{i=1}^{3}(\Delta x_i \Delta z_i)\sum_{i=1}^{3}\Delta y_i + \sum_{i=1}^{3}(\Delta y_i \Delta z_i)\sum_{i=1}^{3}\Delta x_i\Big].\end{aligned}$$

Given these three formulas it is trivial to derive all other integrals for a fourth-order scheme. For example to obtain the formulation for $\mathcal{L}^{s}_{2,0,0}$ one may substitute Δx_i for Δy_i in the formulation for $\mathcal{L}^{s}_{1,1,0}$.

4.3.2 Three-Dimensional Flux through Top and Bottom Hyperplanes

The evaluation of the flux through a bottom BCE requires the integration over a triangular bipyramid. This integration has to be broken up into two parts. There are multiple methods to break up a bipyramid and in this investigation they where divided into an outer tetrahedral and an inner tetrahedral. To generalize the integration let points $1, 2$, and 3 be the vertices of the shared face, point 4 is the cell center of the neighboring cell and point 5 is the cell center of central cell. With this labeling the inner tetrahedral consists of points 1, 2, 3, and 5 and the outer tetrahedral consists of points 1, 2, 3, and 4. When integrating over the bottom BCE, the Taylor series is expanded from the solution point associated with the neighboring cell, point $(j_r; t^{n-1/2})$. When integrating over the top BCE, the Taylor series is expanded from the solution point of the central cell $(j; t^n)$. Since the geometry of the top and bottom BCEs is the same, only the derivation for the bottom BCE is reported here.

To evaluate the integral in Eq. (4.3) over the bottom and top BCE the normal is required and is found using

$$\hat{N}_{out} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} & \hat{t} \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 & t_2 - t_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 & t_3 - t_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 & t_4 - t_1 \end{vmatrix} = - \begin{vmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \end{vmatrix} \hat{t}, \tag{4.16}$$

and

$$\hat{N}_{in} = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} & \hat{t} \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 & t_2 - t_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 & t_3 - t_1 \\ x_5 - x_1 & y_5 - y_1 & z_5 - z_1 & t_5 - t_1 \end{vmatrix} = - \begin{vmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ x_5 - x_1 & y_5 - y_1 & z_5 - z_1 \end{vmatrix} \hat{t}. \tag{4.17}$$

It should be noted that the normals in the i, j and k direction are all zero. Applying Eq. (4.16) to Eq. (4.3) yields:

$$\begin{aligned} \left[(F_i)^{3D}_{bottom}\right]^{n-1/2}_{j_r} =& \sum_{a=0}^{N} \sum_{b=0}^{N-a} \sum_{c=0}^{N-a-b} \left(\frac{\partial^{a+b+c} u_i}{\partial x^a \partial y^b \partial z^c} \right)^{n-1/2}_{j_r} \frac{1}{a!b!c!} \\ & \left[n^t_{in} \int_{V^{4D}_{Bin}} \left(x - x_j\right)^a \left(y - y_j\right)^b \left(z - z_j\right)^c dV^{4D}_{Bin} + \right. \\ & \left. n^t_{out} \int_{V^{4D}_{Bout}} \left(x - x_j\right)^a \left(y - y_j\right)^b \left(z - z_j\right)^c dV^{4D}_{Bout} \right], \end{aligned} \tag{4.18}$$

where $n^t_{in,out}$ are the normal components in the t direction for the inner and outer trapezoid. It should be noted that there is no time integration in Eq. (4.18) because all points inside of the volume V^{4D}_B are at the same time step. To integrate Eq. (4.18) the coordinates are casted into its parametric form:

$$
\begin{aligned}
\left[(F_i)^{3D}_{bottom}\right]^{n-1/2}_{j_r} &= \sum_{a=0}^{N}\sum_{b=0}^{N-a}\sum_{c=0}^{N-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)^{n-1/2}_{j_r}\frac{1}{a!b!c!} \\
&\Bigg[\left(N^t_{in}\right)_{j_r}\int_0^1\int_0^{1-\xi}\int_0^{1-\xi-\eta}\Big(\left[(x_2-x_1)\,\xi+(x_3-x_1)\,\eta+(x_4-x_1)\,\varphi+x_1-x_j\right]^a \\
&\left[(y_2-y_1)\,\xi+(y_3-y_1)\,\eta+(y_4-y_1)\,\varphi+y_1-y_j\right]^b \\
&\left[(z_2-z_1)\,\xi+(z_3-z_1)\,\eta+(z_4-z_1)\,\varphi+z_1-z_j\right]^c\Big)d\varphi d\eta d\xi + \\
&\left(N^t_{out}\right)_{j_r}\int_0^1\int_0^{1-\xi}\int_0^{1-\xi-\eta}\Big(\left[(x_2-x_1)\,\xi+(x_3-x_1)\,\eta+(x_5-x_1)\,\varphi+x_1-x_j\right]^a \\
&\left[(y_2-y_1)\,\xi+(y_3-y_1)\,\eta+(y_5-y_1)\,\varphi+y_1-y_j\right]^b \\
&\left[(z_2-z_1)\,\xi+(z_3-z_1)\,\eta+(z_5-z_1)\,\varphi+z_1-z_j\right]^c\Big)d\varphi d\eta d\xi\Bigg],
\end{aligned}
\tag{4.19}
$$

where $N^t = |J|n^t$ and $|J|$ is the determinate of the coordinate transformation matrix:

$$
J = \begin{pmatrix} x_{2'}-x_{1'} & x_{3'}-x_{1'} & x_{5'}-x_{1'} \\ y_{2'}-y_{1'} & y_{3'}-y_{1'} & y_{5'}-y_{1'} \\ z_{2'}-z_{1'} & z_{3'}-x_{1'} & z_{5'}-z_{1'} \end{pmatrix}
$$

This equation is applied for each BCE in the CCE. For a trapezoidal mesh there are four BCEs. For easy reference this equation is written as
[Eq. (4.19)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_j, u_{jcl})$. Equation (4.19) is easily modified for the top surface by using the Taylor series expanded from the current solution point $(j;n)$.

The resulting equation is:

$$\begin{aligned}
&\left[(F_i)^{3D}_{top}\right]^n_j = \sum_{a=0}^{N}\sum_{b=0}^{N-a}\sum_{c=0}^{N-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)^n_j \frac{1}{a!b!c!} \\
&\Bigg[\left(N^t_{in}\right)_j \int_0^1\int_0^{1-\xi}\int_0^{1-\xi-\eta}\Big(\left[(x_2-x_1)\,\xi+(x_3-x_1)\,\eta+(x_4-x_1)\,\varphi+x_1-x_j\right]^a \\
&\left[(y_2-y_1)\,\xi+(y_3-y_1)\,\eta+(y_4-y_1)\,\varphi+y_1-y_j\right]^b \\
&\left[(z_2-z_1)\,\xi+(z_3-z_1)\,\eta+(z_4-z_1)\,\varphi+z_1-z_j\right]^c\Big)d\varphi d\eta d\xi + \\
&\left(N^t_{out}\right)_j \int_0^1\int_0^{1-\xi}\int_0^{1-\xi-\eta}\Big(\left[(x_2-x_1)\,\xi+(x_3-x_1)\,\eta+(x_5-x_1)\,\varphi+x_1-x_j\right]^a \\
&\left[(y_2-y_1)\,\xi+(y_3-y_1)\,\eta+(y_5-y_1)\,\varphi+y_1-y_j\right]^b \\
&\left[(z_2-z_1)\,\xi+(z_3-z_1)\,\eta+(z_5-z_1)\,\varphi+z_1-z_j\right]^c\Big)d\varphi d\eta d\xi\Bigg],
\end{aligned} \tag{4.20}$$

For future reference let Eq. (4.20) be referenced as:

[Eq. (4.20)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_j, u_{icl})$. An algorithm detailing the numerical procedure used to calculate the flux through the bottom and top surfaces of the space-time element are detailed in Algorithms 8 and 9 respectively. Integrating Eq. (4.19) is easily accomplished through a symbolic mathematical package. The integral in Eqs (4.19) and (4.20) is integrated for select values of a, b, and c. To start let $\Delta x_i = x_i - x_j$,

$\Delta y_i = y_i - y_j$, $\Delta z_i = z_i - z_j$ for $i = 1, 2, 3, 4$. With this definition the integral becomes:

$$\mathcal{L}^{bt3D}_{a,b,c} = \int_0^1 \int_0^{1-\xi} \int_0^{1-\xi-\eta} \prod_{j=1}^{3} \left[\Delta\Psi_2^j \xi + \Delta\Psi_3^j \eta + \Delta\Psi_4^j \varphi + \Delta\Psi_1^j \left(1 - \xi - \eta - w\right)\right]^{\omega^j} d\varphi d\eta d\xi,$$

where $\Psi = \{x, y, z\}$ and $\omega = \{a, b, c\}$. When $a = b = c = 0$ the integration is

$$\mathcal{L}^{bt3D}_{0,0,0} = \frac{1}{6}.$$

When $a = 1$ and $b = c = 0$ the integration is

$$\mathcal{L}^{bt3D}_{1,0,0} = \frac{1}{24} \sum_{i=1}^{4} \Delta x_i.$$

When $a = b = 1$ and $c = 0$

$$\mathcal{L}^{bt3D}_{1,1,0} = \frac{1}{120} \left[\sum_{i=1}^{4} (\Delta x_i \Delta y_i) + \sum_{i=1}^{4} \Delta x_i \sum_{i=1}^{4} \Delta y_i \right].$$

When $a = b = c = 1$

$$\mathcal{L}^{bt3D}_{1,1,1} = \frac{1}{720} \Big[\sum_{i=1}^{4} \Delta x_i \sum_{i=1}^{4} \Delta y_i \sum_{i=1}^{4} \Delta z_i + 2 \sum_{i=1}^{4} (\Delta x_i \Delta y_i \Delta z_i) + \sum_{i=1}^{4} (\Delta x_i \Delta y_i) \sum_{i=1}^{4} \Delta z_i + \sum_{i=1}^{4} (\Delta x_i \Delta z_i) \sum_{i=1}^{4} \Delta y_i + \sum_{i=1}^{4} (\Delta y_i \Delta z_i) \sum_{i=1}^{4} \Delta x_i \Big].$$

The other integrals required for a fourth-order scheme are easily obtained through substituting. For example $\mathcal{L}^{bt}_{0,1,2}$ is obtained by substituting Δz_i for Δx_i in the formulation of $\mathcal{L}^{bt}_{1,1,1}$.

4.3.3 Three-Dimensional Even Derivatives Terms

In the previous two subsections the procedure required to integrate the primary variable over the top, bottom and side BCEs, was detailed. In this subsection the equation used to integrate u_i and its even derivatives is detailed. To determine u_i at the next time step Eqs (4.14), (4.19), and (4.20) need to be evaluated and summed up over every surrounding cell.

$$\sum_{r=0}^{N_B} \left[(F_i)^{3D}_{top}\right]^n_j + \left[(F_i)^{3D}_{bottom}\right]^{n-1/2}_{jr} + \left[(F_i)^{3D}_{side}\right]^{n-1/2}_{jr} = 0,$$

where N_B is the number of cells neighboring cell j. Since the definition of the CE is the same as the original scheme the integration of the linear terms over the top surface becomes null. This allow for an explicit formulation of u_i

$$(u_i)^n_j = -\frac{1}{V_{topCE}} \sum_{r=0}^{N_B} \left(\left[\left(\tilde{F}_i\right)^{3D}_{top}\right]^n_j + \left[(F_i)^{3D}_{bottom}\right]^{n-1/2}_{jr} + \left[(F_i)^{3D}_{side}\right]^{n-1/2}_{jr} \right),$$

where

$$V_{topCE} = \frac{1}{6} \sum_{r=0}^{N_B} \left(\left(N^t_{in}\right)_{jr} + \left(N^t_{out}\right)_{jr} \right)$$

and

$$\left[\left(\tilde{F}_i\right)^{3D}_{top}\right]^n_{j_r} = \sum_{a=0}^{N}\sum_{b=0}^{N-a}\sum_{c=0}^{N-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)^n_j \frac{1}{a!b!c!}$$

$$\Bigg[\left(N^t_{in}\right)_{j_r}\int_0^1\int_0^{1-\xi}\int_0^{1-\xi-\eta}\Big(\big[(x_2-x_1)\,\xi+(x_3-x_1)\,\eta+(x_4-x_1)\,\varphi+x_1-x_j\big]^a$$

$$\big[(y_2-y_1)\,\xi+(y_3-y_1)\,\eta+(y_4-y_1)\,\varphi+y_1-y_j\big]^b$$

$$\big[(z_2-z_1)\,\xi+(z_3-z_1)\,\eta+(z_4-z_1)\,\varphi+z_1-z_j\big]^c\Big)d\varphi d\eta d\xi+$$

$$\left(N^t_{out}\right)_{j_r}\int_0^1\int_0^{1-\xi}\int_0^{1-\xi-\eta}\Big(\big[(x_2-x_1)\,\xi+(x_3-x_1)\,\eta+(x_5-x_1)\,\varphi+x_1-x_j\big]^a$$

$$\big[(y_2-y_1)\,\xi+(y_3-y_1)\,\eta+(y_5-y_1)\,\varphi+y_1-y_j\big]^b$$

$$\big[(z_2-z_1)\,\xi+(z_3-z_1)\,\eta+(z_5-z_1)\,\varphi+z_1-z_j\big]^c\Big)d\varphi d\eta d\xi\Bigg],$$

with $(a,b,c) \neq \big\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\big\}$. Essentially the above equation is Eq. (4.20) without the $(u_i)^n_j$, $(u_{i,x})^n_j$, $(u_{i,y})^n_j$, $(u_{i,z})^n_j$ terms.

Next it will be shown that the same steps are used to determine all of the even derivatives. Since each even derivative is expressed as a Taylor series and the definition of the CE and SE remain the same the same integrals are reused. In fact the only difference between this integration and the previous one is the order of the Taylor series to be integrated. Since the examples used in this chapter are fourth-order accurate this derivation will focus on the integration of the second derivatives in the context of a fourth-order scheme. To proceed, Eq. (4.12) is cast into into the following divergence-free equations:

$$
\begin{aligned}
&\nabla \cdot \mathbf{h}_{i,xx} = 0, \quad \nabla \cdot \mathbf{h}_{i,xy} = 0, \quad \nabla \cdot \mathbf{h}_{i,xz} = 0,\\
&\nabla \cdot \mathbf{h}_{i,yx} = 0, \quad \nabla \cdot \mathbf{h}_{i,yy} = 0, \quad \nabla \cdot \mathbf{h}_{i,yz} = 0,\\
&\nabla \cdot \mathbf{h}_{i,zx} = 0, \quad \nabla \cdot \mathbf{h}_{i,zy} = 0, \quad \nabla \cdot \mathbf{h}_{i,zz} = 0,
\end{aligned}
\tag{4.21}
$$

where

$$
\begin{aligned}
\mathbf{h}_{i,xx} &= (f^x_{i,xx}, f^y_{i,xx}, f^z_{i,xx}, u_{i,xx}), \quad & \mathbf{h}_{i,xy} &= (f^x_{i,xy}, f^y_{i,xy}, f^z_{i,xy}, u_{i,xy}),\\
\mathbf{h}_{i,xz} &= (f^x_{i,xz}, f^y_{i,xz}, f^z_{i,xz}, u_{i,xz}), \quad & \mathbf{h}_{i,yx} &= (f^x_{i,yx}, f^y_{i,yx}, f^z_{i,yx}, u_{i,yx}),\\
\mathbf{h}_{i,yy} &= (f^x_{i,yy}, f^y_{i,yy}, f^z_{i,yy}, u_{i,yy}), \quad & \mathbf{h}_{i,yz} &= (f^x_{i,yz}, f^y_{i,yz}, f^z_{i,yz}, u_{i,yz}),\\
\mathbf{h}_{i,zx} &= (f^x_{i,zx}, f^y_{i,zx}, f^z_{i,zx}, u_{i,zx}), \quad & \mathbf{h}_{i,zy} &= (f^x_{i,zy}, f^y_{i,zy}, f^z_{i,zy}, u_{i,zy}),\\
\mathbf{h}_{i,zz} &= (f^x_{i,zz}, f^y_{i,zz}, f^z_{i,zz}, u_{i,zz}).
\end{aligned}
\tag{4.22}
$$

Aided by the Gauss theorem in the four-dimensional space-time domain, the above differential equations are recast into the following integral equations:

$$
\begin{aligned}
&\oint \mathbf{h}_{i,xx} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,xy} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,xz} \cdot d\mathbf{s} = 0,\\
&\oint \mathbf{h}_{i,yx} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,yy} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,yz} \cdot d\mathbf{s} = 0,\\
&\oint \mathbf{h}_{i,zx} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,zy} \cdot d\mathbf{s} = 0, \quad \oint \mathbf{h}_{i,zz} \cdot d\mathbf{s} = 0.
\end{aligned}
\tag{4.23}
$$

The space-time integration procedure is applied nine times to enforce flux conservation over all second derivatives. In the context of a fourth-fourth order scheme each second derivatives is expressed as a first-order Taylor series:

$$\begin{aligned}
u^*_{i,xx} &= u_{i,xx} + u_{i,xxx}\Delta x + u_{i,xxy}\Delta y + u_{i,xxz}\Delta z + u_{i,xxt}\Delta t\\
u^*_{i,xy} &= u_{i,xy} + u_{i,xyx}\Delta x + u_{i,xyy}\Delta y + u_{i,xyz}\Delta z + u_{i,xyt}\Delta t\\
u^*_{i,xz} &= u_{i,xz} + u_{i,xzx}\Delta x + u_{i,xzy}\Delta y + u_{i,xzz}\Delta z + u_{i,xzt}\Delta t\\
u^*_{i,yx} &= u_{i,yx} + u_{i,yxx}\Delta x + u_{i,yxy}\Delta y + u_{i,yxz}\Delta z + u_{i,yxt}\Delta t\\
u^*_{i,yy} &= u_{i,yy} + u_{i,yyx}\Delta x + u_{i,yyy}\Delta y + u_{i,yyz}\Delta z + u_{i,yyt}\Delta t\\
u^*_{i,yz} &= u_{i,yz} + u_{i,yzx}\Delta x + u_{i,yzy}\Delta y + u_{i,yzz}\Delta z + u_{i,yzt}\Delta t\\
u^*_{i,zx} &= u_{i,zx} + u_{i,zxx}\Delta x + u_{i,zxy}\Delta y + u_{i,zxz}\Delta z + u_{i,zxt}\Delta t\\
u^*_{i,zy} &= u_{i,zy} + u_{i,zyx}\Delta x + u_{i,zyy}\Delta y + u_{i,zyz}\Delta z + u_{i,zyt}\Delta t\\
u^*_{i,zz} &= u_{i,zz} + u_{i,zzx}\Delta x + u_{i,zzy}\Delta y + u_{i,zzz}\Delta z + u_{i,zzt}\Delta t
\end{aligned} \tag{4.24}$$

The associated fluxes are also expressed as a first-order Taylor series:

$$\begin{aligned}
(f_{i,xx}^{x,y,z})^* &= f_{i,xx}^{x,y,z} + f_{i,xxx}^{x,y,z}\Delta x + f_{i,xxy}^{x,y,z}\Delta y + f_{i,xxz}^{x,y,z}\Delta z + f_{i,xxt}^{x,y,z}\Delta t \\
(f_{i,xy}^{x,y,z})^* &= f_{i,xy}^{x,y,z} + f_{i,xyx}^{x,y,z}\Delta x + f_{i,xyy}^{x,y,z}\Delta y + f_{i,xyz}^{x,y,z}\Delta z + f_{i,xyt}^{x,y,z}\Delta t \\
(f_{i,xz}^{x,y,z})^* &= f_{i,xz}^{x,y,z} + f_{i,xzx}^{x,y,z}\Delta x + f_{i,xzy}^{x,y,z}\Delta y + f_{i,xzz}^{x,y,z}\Delta z + f_{i,xzt}^{x,y,z}\Delta t \\
(f_{i,yx}^{x,y,z})^* &= f_{i,yx}^{x,y,z} + f_{i,yxx}^{x,y,z}\Delta x + f_{i,yxy}^{x,y,z}\Delta y + f_{i,yxz}^{x,y,z}\Delta z + f_{i,yxt}^{x,y,z}\Delta t \\
(f_{i,yy}^{x,y,z})^* &= f_{i,yy}^{x,y,z} + f_{i,yyx}^{x,y,z}\Delta x + f_{i,yyy}^{x,y,z}\Delta y + f_{i,yyz}^{x,y,z}\Delta z + f_{i,yyt}^{x,y,z}\Delta t \\
(f_{i,yz}^{x,y,z})^* &= f_{i,yz}^{x,y,z} + f_{i,yzx}^{x,y,z}\Delta x + f_{i,yzy}^{x,y,z}\Delta y + f_{i,yzz}^{x,y,z}\Delta z + f_{i,yzt}^{x,y,z}\Delta t \\
(f_{i,zx}^{x,y,z})^* &= f_{i,zx}^{x,y,z} + f_{i,zxx}^{x,y,z}\Delta x + f_{i,zxy}^{x,y,z}\Delta y + f_{i,zxz}^{x,y,z}\Delta z + f_{i,zxt}^{x,y,z}\Delta t \\
(f_{i,zy}^{x,y,z})^* &= f_{i,zy}^{x,y,z} + f_{i,zyx}^{x,y,z}\Delta x + f_{i,zyy}^{x,y,z}\Delta y + f_{i,zyz}^{x,y,z}\Delta z + f_{i,zyt}^{x,y,z}\Delta t \\
(f_{i,zz}^{x,y,z})^* &= f_{i,zz}^{x,y,z} + f_{i,zzx}^{x,y,z}\Delta x + f_{i,zzy}^{x,y,z}\Delta y + f_{i,zzz}^{x,y,z}\Delta z + f_{i,zzt}^{x,y,z}\Delta t
\end{aligned} \tag{4.25}$$

Since the procedure is the same for all second derivatives $u_{i,xy}$ will be used as an example. The integration over the side surface for $u_{i,xy}$ is expressed as

$$\begin{aligned}
\left[(F_{i,xy})_{side}^{3D}\right]_{j_r}^{n-1/2} &= \sum_{Si}^{1,2,3}\sum_{x^i}^{x,y,z} \\
&\left[N_{x^i}\frac{\Delta t}{2}\left(0.5 f_{i,xyt}^{x^i}\frac{\Delta t}{2} + 0.5 f_{i,xy}^{x^i} + \frac{1}{6} f_{i,xyx}^{x^i}\left(\Delta x_1 + \Delta x_2 + \Delta x_3\right) + \right.\right. \\
&\left.\left.\frac{1}{6} f_{i,xyy}^{x^i}\left(\Delta y_1 + \Delta y_2 + \Delta y_3\right) + \frac{1}{6} f_{i,xyz}^{x^i}\left(\Delta z_1 + \Delta z_2 + \Delta z_3\right)\right)\right]_{j_r}^{n-1/2}
\end{aligned} \tag{4.26}$$

The flux over the bottom BCE for $u_{i,xy}$ is:

$$\begin{aligned}\left[(F_{i,xy})^{3D}_{bottom}\right]^{n-1/2}_{j_r} &= \left(N^t_{in}\right)_{j_r}\left[\frac{u_{i,xy}}{6}+\frac{u_{i,xyx}}{24}\left(\Delta x_1+\Delta x_2+\Delta x_3+\Delta x_4\right)+\right.\\
&\left.\frac{u_{i,xyy}}{24}\left(\Delta y_1+\Delta y_2+\Delta y_3+\Delta y_4\right)+\frac{u_{i,xyz}}{24}\left(\Delta z_1+\Delta z_2+\Delta z_3+\Delta z_4\right)\right]^{n-1/2}_{j_r}+\\
&\left(N^t_{out}\right)_{j_r}\left[\frac{u_{i,xy}}{6}+\frac{u_{i,xyx}}{24}\left(\Delta x_1+\Delta x_2+\Delta x_3+\Delta x_5\right)+\right.\\
&\left.\frac{u_{i,xyy}}{24}\left(\Delta y_1+\Delta y_2+\Delta y_3+\Delta y_5\right)+\frac{u_{i,xyz}}{24}\left(\Delta z_1+\Delta z_2+\Delta z_3+\Delta z_5\right)\right]^{n-1/2}_{j_r}.\end{aligned} \tag{4.27}$$

$$\begin{aligned}\left[(F_{i,xy})^{3D}_{top}\right]^{n}_{j_r} &= \left(N^t_{in}\right)_{j_r}\left[\frac{u_{i,xy}}{6}+\frac{u_{i,xyx}}{24}\left(\Delta x_1+\Delta x_2+\Delta x_3+\Delta x_4\right)+\right.\\
&\left.\frac{u_{i,xyy}}{24}\left(\Delta y_1+\Delta y_2+\Delta y_3+\Delta y_4\right)+\frac{u_{i,xyz}}{24}\left(\Delta z_1+\Delta z_2+\Delta z_3+\Delta z_4\right)\right]^{n}_{j_r}+\\
&\left(N^t_{out}\right)_{j_r}\left[\frac{u_{i,xy}}{6}+\frac{u_{i,xyx}}{24}\left(\Delta x_1+\Delta x_2+\Delta x_3+\Delta x_5\right)+\right.\\
&\left.\frac{u_{i,xyy}}{24}\left(\Delta y_1+\Delta y_2+\Delta y_3+\Delta y_5\right)+\frac{u_{i,xyz}}{24}\left(\Delta z_1+\Delta z_2+\Delta z_3+\Delta z_5\right)\right]^{n}_{j_r}.\end{aligned} \tag{4.28}$$

To calculate $(u_{i,xy})^n_j$ at the new time step the flux over all BCEs associated with cell j are summed together to obtain

$$\left(u_{i,xy}\right)^n_j = -\frac{1}{V_{topCE}}\sum_{r=0}^{N_B}\left(\left[\left(F_{i,xy}\right)^{3D}_{bottom}\right]^{n-1/2}_{j_r}+\left[\left(F_{i,xy}\right)^{3D}_{side}\right]^{n-1/2}_{j_r}\right).$$

It is important to note that all of the terms on the RHS are from the previous time step, which makes this equation completely explicit. Just as it was with the one- and two-dimensional CESE schemes, the CESE scheme is completely explicit as long as the unknowns are calculated in the correct order.

4.4 Three-Dimensional Central Differencing Procedure

In this section the central differencing procedure used to find the odd derivatives at the new time step is illustrated. There are two primary methods to calculate the odd derivatives, the CFL insensitive $c-\tau$ scheme[59, 61] and the edge-based derivative (EBD) scheme [44]. Both method use a similar approach, only differing in how they determine the locations to evaluate the Taylor series expansion at. TO proceed, this section is split into two subsections. The first derives an arbitrary order $c-\tau$ scheme and the second derives the arbitrary order EBD scheme. Both of these sections begin by deriving the core scheme which calculates the first derivatives and then follows up with a generalization to higher derivatives.

4.4.1 Three-Dimensional $c-\tau$ Scheme

As a preface the necessary geometry will be explained. First consider the current cell at location j_0 and the surrounding cells denoted by j_r for $r = 1, 2, \ldots, N_b$, where N_b is the number of neighbors. For a tetrahedral mesh N_b would be 4. These cells exist at both the current time step, n, and the previous time step $n - 1/2$ giving them the space-time location of (j_r, n) $r = 0, \ldots, N_b$ for the current time step and $(j_r, n - 1/2)$ $r = 0, \ldots, N_b$ for the previous time step. There are two important locations in each cell, (i) the solution point denoted by $j_r^\times$, (ii) the Taylor expansion point denoted by j_r^+. The derivation below can be applied to any type of mesh, i.e. tetrahedral, hexagonal, prism, and four-sided pyramid, but for simplicity the derivation will be restricted to tetrahedral meshes. In what follows is an outline detailing each step required to calculate the first derivatives using the $c-\tau$ algorithm.

The first step is to determine the location of the Taylor series expansion points, which is accomplished through the $c-\tau$ scheme detailed in[61].

$$x_{j_r}^+ = 0.5(x_0 + x_{j_r}) + 0.5(x_{j_r} - x_0^\times)CFL$$

where CFL is the local CFL number associated with cell $(j_0, n-1/2)$. The next step is to write a Taylor series expansions from $(j_0^\times, n)$ to each (j_r^+, n) $r = 1, \ldots, N_b$.

$$\begin{aligned}\left[u_i^*(j_r^+, n)\right]_{j_0^\times}^n = &\sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^n \\ &\frac{\left(x_{j_r^+} - x_{j_0^\times}\right)^a \left(y_{j_r^+} - y_{j_0^\times}\right)^b \left(z_{j_r^+} - z_{j_0^\times}\right)^c}{a!b!c!}.\end{aligned} \tag{4.29}$$

By applying Eq. (4.29) to each of the surrounding cells N_b equations with three$+N_b$ unknowns per governing equation are obtained. The unknowns are the first derivatives of the conserved variables and the values of the conserved variables at the expansion points, $\left[u_i^*(j^+, n)\right]_{j_0^\times}^n$ $j = 1, \ldots, N_b$.

Next the unknowns at $\left[u_i^*(j_r^+, n)\right]_{j_0^\times}^n$ are approximated by a Taylor series expansion from $j_r^\times$ at the previous half time step to j_r^+ at the current time step, i.e.

$\left[u_i^*(j_r^+,n)\right]_{j_0^\times}^n \approx \left[u_i'(j_r^+,n)\right]_{j_r^\times}^{n-1/2}$ where

$$\left[u_i'(j^+,n)\right]_{j_r^\times}^{n-1/2} = \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\sum_{d=0}^{A-a-b-c}\left(\frac{\partial^B u_i}{\partial x^a \partial y^b \partial z^c \partial t^d}\right)_{j_r^\times}^{n-\frac{1}{2}} \frac{\left(x_{j_r^+}-x_{j_r^\times}\right)^a \left(y_{j_r^+}-y_{j_r^\times}\right)^b \left(z_{j_r^+}-z_{j_r^\times}\right)^c}{a!b!c!d!}\left(\frac{\Delta t}{2}\right)^d \tag{4.30}$$

Substituting Eq. (4.30) into Eq. (4.29) and moving the unknowns to the LHS yields:

$$\left[u_{i,x}\Delta x_{j_r^+} + u_{i,y}\Delta y_{j_r^+} + u_{i,z}\Delta z_{j_r^+}\right]_{j_0^\times}^n = \left[u_i'(j^+,n)\right]_{j^\times}^{n-1/2} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^n \frac{\left(\Delta x_{j_r^+}\right)^a \left(\Delta y_{j_r^+}\right)^b \left(\Delta z_{j_r^+}\right)^c}{a!b!c!}, \tag{4.31}$$

where $\Delta x_{j_r^+} = x_{j_r^+} - x_{j_0^\times}$, $\Delta y_{j_r^+} = y_{j_r^+} - y_{j_0^\times}$, $\Delta z_{j_r^+} = z_{j_r^+} - z_{j_0^\times}$ and $(a,b,c) \neq (1,0,0),(0,1,0),(0,0,1)$.

By applying Eq. (4.31) to each neighboring cells an overdetermined system of equation consisting of three unknowns and N_b equations is formed. This is advantageous because it allows for multiple solutions to be determined and then select the most appropriate solution through a weighting procedure. To determine the possible solutions the equations listed in Eq. (4.31) are grouped into sets of three. For example a tetrahedral mesh has four surrounding cells and would be grouped as $[(1,2,3),(1,2,4),(1,3,4),(2,3,4)]$. Taking the first triplet from the set results in the following equation:

$$\begin{bmatrix} \Delta x_{1+} & \Delta y_{1+} & \Delta z_{1+} \\ \Delta x_{2+} & \Delta y_{2+} & \Delta z_{2+} \\ \Delta x_{3+} & \Delta y_{3+} & \Delta z_{3+} \end{bmatrix} \begin{bmatrix} (u_i)_x^{(1)} \\ (u_i)_y^{(1)} \\ (u_i)_z^{(1)} \end{bmatrix} =$$

$$\begin{bmatrix} \left[u_i'(1^+,n)\right]_{1^\times}^{n-1/2} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^{n} \frac{\left(\Delta x_{1+}\right)^a\left(\Delta y_{1+}\right)^b\left(\Delta z_{1+}\right)^c}{a!b!c!} \\ \left[u_i'(2^+,n)\right]_{2^\times}^{n-1/2} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^{n} \frac{\left(\Delta x_{2+}\right)^a\left(\Delta y_{2+}\right)^b\left(\Delta z_{2+}\right)^c}{a!b!c!} \\ \left[u_i'(3^+,n)\right]_{3^\times}^{n-1/2} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^{n} \frac{\left(\Delta x_{3+}\right)^a\left(\Delta y_{3+}\right)^b\left(\Delta z_{3+}\right)^c}{a!b!c!} \end{bmatrix}$$

$$\text{for } (a,b,c) \neq (0,1,0),(1,0,0),(0,0,1). \tag{4.32}$$

Using Eq. (4.32) on all groupings of equations will generate multiple solutions to $u_{i,x}$, $u_{i,y}$ and $u_{i,z}$. The superscript (1) for the unknowns in Eq. (4.32) represents the first possible solution for $u_{i,x}$, $u_{i,y}$ and $u_{i,z}$.

The final step is to determine the smoothest solution through a weighting algorithm. Any of the previously derived weighting schemes are applicable to the higher order CESE method without modification. In the present investigation the W2[61], S2[72] and Eq. (3.49) schemes are used.

This same procedure is easily adapted to determine any of the odd derivatives. For example to find the variables $u_{i,xyx}$, $u_{i,xyy}$, and $u_{i,xyz}$ consider the Taylor series expansion of $u_{i,xy}$ from $j_0^{\times}$ to j_r^{+}

$$u^*_{i,xy} = \left[u_{i,xy} + u_{i,xyx}\Delta x_{j^+} + u_{i,xyy}\Delta y_{j^+} + u_{i,xyz}\Delta z_{j^+}\right]^n_{j_0} + \\ \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b}\left(\frac{\partial^{2+a+b+c}u_i}{\partial x^{a+1}\partial y^{b+1}\partial z^{c+1}}\right)^n_{j_0^{\times}}\frac{\Delta x^a}{a!}\frac{\Delta y^b}{b!}\frac{\Delta z^b}{c!} \tag{4.33}$$

for $a, b, c \neq \left\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\right\}$.

The first four Taylor series coefficients are listed explicitly. This is important because the alternative rule for differentiation is not used and this will decrease the ambiguity for the terms listed in the summation.

An expression for the Taylor series expansion from the previous time step to the current one is:

$$\left[(u_{i,xy})^n_{j_r^+}\right]^{n-1/2}_{j_r^{\times}} = \\ \left[u_{i,xy} + u_{i,xyx}\left(\Delta x_{j_r^+}\right) + u_{i,xyy}\left(\Delta y_{j_r^+}\right) + u_{i,xyz}\left(\Delta z_{j_r^+}\right) + u_{i,xyt}\frac{\Delta t}{2}\right]^{n-1/2}_{j_r^{\times}} + \\ \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b-c}\sum_{d=0}^{A-a-b-c}\left(\frac{\partial^{2+a+b+c+d}u_i}{\partial x^{a+1}\partial y^{b+1}\partial z^{c}\partial t^{d}}\right)^{n-1/2}_{j_r^{\times}}\frac{\left(\Delta x_{j_r^+}\right)^a}{a!}\frac{\left(\Delta y_{j_r^+}\right)^b}{b!}\frac{\left(\Delta z_{j_r^+}\right)^c}{c!}\frac{\Delta t^d}{2d!} \tag{4.34}$$

for $(a, b, c, d) \neq \left\{(0,0,0,0),(1,0,0,0),(0,1,0,0),(0,0,1,0)\right\}$.

By applying the same procedure to $u^*_{i,xy}$ that was used for u^*_i a solution for $u_{i,xyx}$,

$u_{i,xyy}$, and $u_{i,xyz}$ is derived:

$$\begin{bmatrix} \Delta x_{1^+} & \Delta y_{1^+} & \Delta z_{1^+} \\ \Delta x_{2^+} & \Delta y_{2^+} & \Delta z_{2^+} \\ \Delta x_{3^+} & \Delta y_{3^+} & \Delta z_{3^+} \end{bmatrix} \begin{bmatrix} (u_i)^{(1)}_{xyx} \\ (u_i)^{(1)}_{xyy} \\ (u_i)^{(1)}_{xyz} \end{bmatrix} = \begin{bmatrix} \left[u'_{i,xy}(1^+, n)\right]^{n-1/2}_{1^\times} - \left(u_{i,xy}\right)^n_{j_0^\times} \\ \left[u'_{i,xy}(2^+, n)\right]^{n-1/2}_{2^\times} - \left(u_{i,xy}\right)^n_{j_0^\times} \\ \left[u'_{i,xy}(3^+, n)\right]^{n-1/2}_{3^\times} - \left(u_{i,xy}\right)^n_{j_0^\times} \end{bmatrix} \tag{4.35}$$

Equation (4.35) represents one possible set of solutions for $u_{i,xyx}$, $u_{i,xyy}$, and $u_{i,xyz}$. When all possible solutions are found they are weighted together to find the vales at the new time step. This procedures is then applied to all of the other third derivatives. Although this procedure was limited to a fourth-order scheme it is easily expandable to any desired order by including higher order Taylor series coefficients.

4.4.2 Three-Dimensional EBD Scheme

As it was with the two-dimensional scheme the EBD method may also be used to compute the odd derivatives. For this method each expansion point is calculated in the same fashion as the two-dimensional scheme. It should be noted that this method has not been developed for cells with four nodes per face. To begin, the expansion points are found via:

$$\hat{N}_i = N_0 + \Omega \left(N_i - N_0\right), \text{ for } i = 1, 2, 3, 4 \tag{4.36}$$

Where N_0 is the centroid of the central tetrahedron and N_i are the vertices of the central tetrahedron, and Ω is an adjustable parameter that controls the amount of dissipation added. As the value of Ω increases the amount of dissipation increases as well. For this investigation the function used to determine Ω was

$$\Omega \equiv Max\left(\Omega_{min}, CFL\Omega_{max}\right).$$

The only requirement of this equation is $\Omega_{min} > 1$.

The next step is to determine which solution points are used to calculate u_i at each expansion point. Following the same procedure used in the two-dimensional EBD scheme, a single solution point is used to calculate u_i at each expansion point. For example, assume that the shared face of cells j_0 and j_1 contains the vertices N_1, N_2, and N_3. This forms a system of equations which is used to determine one possible solution for $u_{i,xyx}$, $u_{i,xyy}$, and $u_{i,xyz}$. This system of equations takes the form:

$$\begin{bmatrix} \Delta x_{\hat{1}} & \Delta y_{\hat{1}} & \Delta z_{\hat{1}} \\ \Delta x_{\hat{2}} & \Delta y_{\hat{2}} & \Delta z_{\hat{2}} \\ \Delta x_{\hat{3}} & \Delta y_{\hat{3}} & \Delta z_{\hat{3}} \end{bmatrix} \begin{bmatrix} (u_{i,x}^{(1)})_{j_0^\times}^n \\ (u_{i,y}^{(1)})_{j_0^\times}^n \\ (u_{i,z}^{(1)})_{j_0^\times}^n \end{bmatrix}$$
$$= \begin{bmatrix} \left[(u_i')_{\hat{1}}^n\right]_{1^\times}^{n-1/2} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b} \left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^n \frac{(\Delta x_{\hat{1}})^a(\Delta y_{\hat{1}})^b(\Delta z_{\hat{1}})^c}{a!b!} \\ \left[(u_i')_{\hat{2}}^n\right]_{1^\times}^{n-1/2} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b} \left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^n \frac{(\Delta x_{\hat{2}})^a(\Delta y_{\hat{2}})^b(\Delta z_{\hat{2}})^c}{a!b!} \\ \left[(u_i')_{\hat{3}}^n\right]_{1^\times}^{n-1/2} - \sum_{a=0}^{A}\sum_{b=0}^{A-a}\sum_{c=0}^{A-a-b} \left(\frac{\partial^{a+b+c}u_i}{\partial x^a \partial y^b \partial z^c}\right)_{j_0^\times}^n \frac{(\Delta x_{\hat{3}})^a(\Delta y_{\hat{3}})^b(\Delta z_{\hat{3}})^c}{a!b!} \end{bmatrix} \tag{4.37}$$

for $(a, b, c) \neq (0, 1, 0), (1, 0, 0), (0, 0, 1)$,

where $\left[(u_i')_{\hat{1}}^n\right]_{1^\times}^{n-1/2}$ is defined by Eq. (4.30). Equation (4.37) is then applied to the other three surrounding cells. This results in four possible solutions for the first derivatives: $(u_{i,x}^{(i)})_{j_0}^n$, $(u_{i,y}^{(i)})_{j_0}^n$, and $(u_{i,z}^{(i)})_{j_0}^n$. These values are than weighted together to obtain the values of $(u_{i,x})_{j_0}^n$, $(u_{i,y})_{j_0}^n$, and $(u_{i,z})_{j_0}^n$ at the new time step.

By applying the same procedure to $u_{i,xy}^*$ that was used for u_i^* a solution for $u_{i,xyx}$, $u_{i,xyy}$, and $u_{i,xyz}$ is derived:

$$\begin{bmatrix} \Delta x_{\hat{1}} & \Delta y_{\hat{1}} & \Delta z_{\hat{1}} \\ \Delta x_{\hat{2}} & \Delta y_{\hat{2}} & \Delta z_{\hat{2}} \\ \Delta x_{\hat{3}} & \Delta y_{\hat{3}} & \Delta z_{\hat{3}} \end{bmatrix} \begin{bmatrix} (u_{i,xyx}^n)^{(1)} \\ (u_{i,xyy}^n)^{(1)} \\ (u_{i,xyz}^n)^{(1)} \end{bmatrix} = \begin{bmatrix} \left[u_{i,xy}'\right]_{\hat{1}}^{n-1/2} - \left(u_{i,xy}\right)_{j_0^\times}^n \\ \left[u_{i,xy}'\right]_{\hat{2}}^{n-1/2} - \left(u_{i,xy}\right)_{j_0^\times}^n \\ \left[u_{i,xy}'\right]_{\hat{3}}^{n-1/2} - \left(u_{i,xy}\right)_{j_0^\times}^n \end{bmatrix} \tag{4.38}$$

Equation (4.38) represents one possible set of solutions for $u_{i,xyx}$, $u_{i,xyy}$, and $u_{i,xyz}$.

When all possible solutions are found they are weighted together to find the vales at the new time step.

This procedures is then applied to all of the other third derivatives. Although this procedure was limited to a fourth-order scheme it is easily expandable to any desired order by including higher order Taylor series coefficients.

4.5 Numerical Procedure

Since both the even- and odd derivatives requires information of the unknowns at the new time step, the order in which the various derivatives are updated is important to ensure that the procedure is explicit. The procedure outline in this section is constrained to a fourth-order scheme but the this method is easily expanded to sixth-, eighth- or even higher-orders of accuracy. For a fourth-order scheme, the order in which the values are calculated is:

1. Calculate the temporal derivatives of the conserved variables, the fluxes, and the spatial and temporal derivatives of the fluxes, at the previous time step.

2. Apply the original second-order CESE scheme to the second derivatives of the unknowns, e.g. $(u_i)_{xx}$, $(u_i)_{xy}$, $(u_i)_{xz}$, $\ldots$, at the new time step.

3. Apply the central difference procedure to calculate the third derivatives of the unknowns, e.g. $(u_i)_{xxx}$, $(u_i)_{xxy}$, $(u_i)_{xxz}$, $\ldots$, at the new time step.

4. Apply the space-time integration procedure to calculate the conserved variables, u_i, at the new time step.

5. Apply the central difference procedure to the first derivatives of the unknowns, to calculate $(u_i)_x,\ (u_i)_y,\ (u_i)_z$ at the new time step.

4.6 Results and Discussions

To assess the accuracy of the three-dimensional, unstructured hyperbolic solver, the following benchmark problems are solved by the newly developed fourth-order, three-dimensional CESE method: (i) a single scalar advection equation to assess the convergence rate, and (ii) the Euler equation for Mach 3 supersonic flow passing over a spherical blunt body.

4.6.1 Advection Equation for Convergence

A single scalar advection equation, Eq. (4.39), is solved by the newly developed fourth-order CESE method with a series mesh refinement to assess the order of convergence of the new scheme.

$$\frac{\partial u}{\partial t} + a_x \frac{\partial u}{\partial x} + a_y \frac{\partial u}{\partial y} + a_z \frac{\partial u}{\partial z} = 0 \tag{4.39}$$

The solution of the above equation assumes the following form

$$u = \sin(a_x x + a_y y + a_z z + a_t t), \tag{4.40}$$

The computational domain is a cube. The length of each side line segment of the cube is 2π. Over all 6 surfaces, periodic boundary conditions are imposed. The wave speeds are set at $a_x = a_y = a_z = 1$ and $a_t = -(a_x^2 + a_y^2 + a_z^2)$. The simulation is run until the non-dimensional time reaches 25. The time period allows the wave to propagate through the domain $25/(2\pi)$ times. To determine the rate of convergence, the L_2 norm is calculated and compared against a characteristic length taken to be $(Volume/ncell)^{-1/3}$, where $ncell$ is the number of cells. The result of the simulation is shown in Fig. (4.4). A best fit line is drawn to show the convergence rate, which is at 4.3 and 2.6 for the fourth- and second-order CESE methods, respectively.

4.6.2 Supersonic Flow over a Spherical Blunt Body

The objective of this case is to assess the capability of the new fourth-order CESE method in dealing with solution discontinuities. To accomplish this, supersonic flow over a sphere is considered. In this simulation the free stream Mach number is 3, the free stream pressure is 1 bar, and the free stream density is 1.23 kg/m^3. The specific heat ratio of the air is 1.4, and the gas constant is 287.15 J/kg K. The radius of the sphere is 0.5 m. The quality of the numerical result is assessed by the post shock density, shock standoff distance and the shock profile.

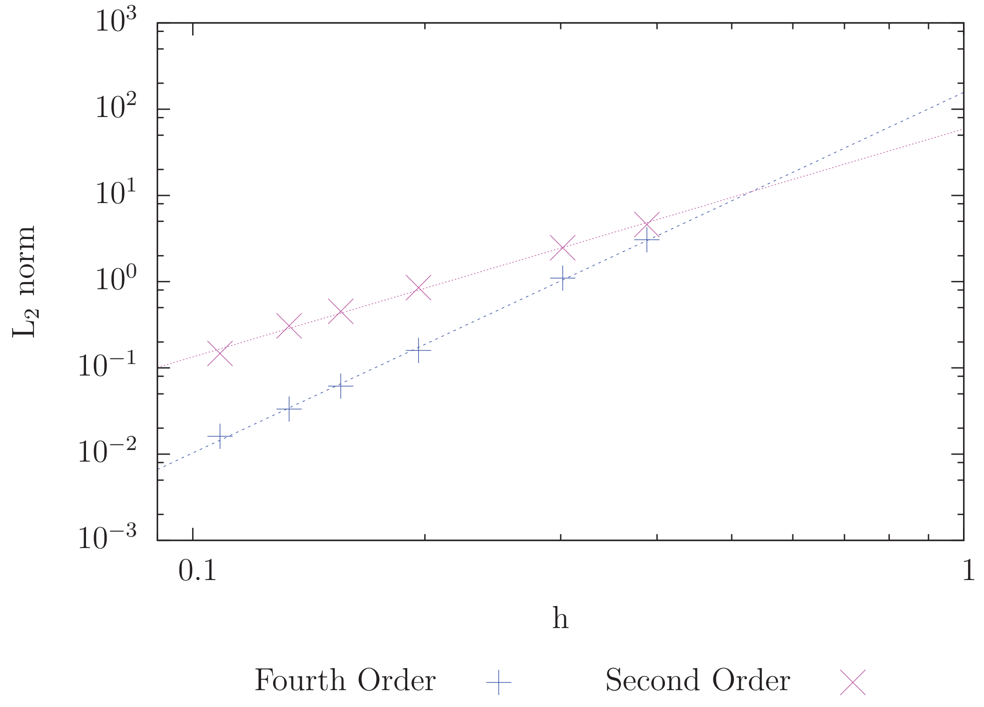


Figure 4.4: Convergence test for the three-dimensional convection equation.

The simulation was run for 5 thousand iterations at an average CFL number of 0.533. The mesh is composed of roughly 4.4 million unstructured tetrahedrals.

The calculated post shock density is 4.613 kg/m^3, which compares favorably to the reported value of 4.744 kg/m^3. The calculated shock standoff distance is about 0.11 m, which compares well with 0.1 m predicted by Ambrosio and Wortman's[77] relation,

$$\Delta = 0.143R\, e^{3.24/M_\infty^2}, \tag{4.41}$$

where R is the radius of the sphere and M_∞ is the free stream Mach number. Billig[78] showed that the shock profile can be represented by the following relationship

$$x = R + \Delta - R_c \cot^2\theta \left[\left(1 + \frac{y^2 \tan^2\theta}{R_c^2} \right)^{0.5} - 1 \right], \tag{4.42}$$

where Δ is the shock-stand-off distance given by Eq. (4.41), θ is the Mach angle which is equal to $\sin^{-1}(1/M_\infty)$, and R_c is the radius of the curvature which is equal to

$$R_c = 1.143R\, e^{(0.54/(M_\infty - 1)^{1.2}}.$$

A numerical Schlieren image constructed by taking the gradient of density shown in Fig. (4.5). Points in the figure show the shock profile generated by Eq. (4.42). The

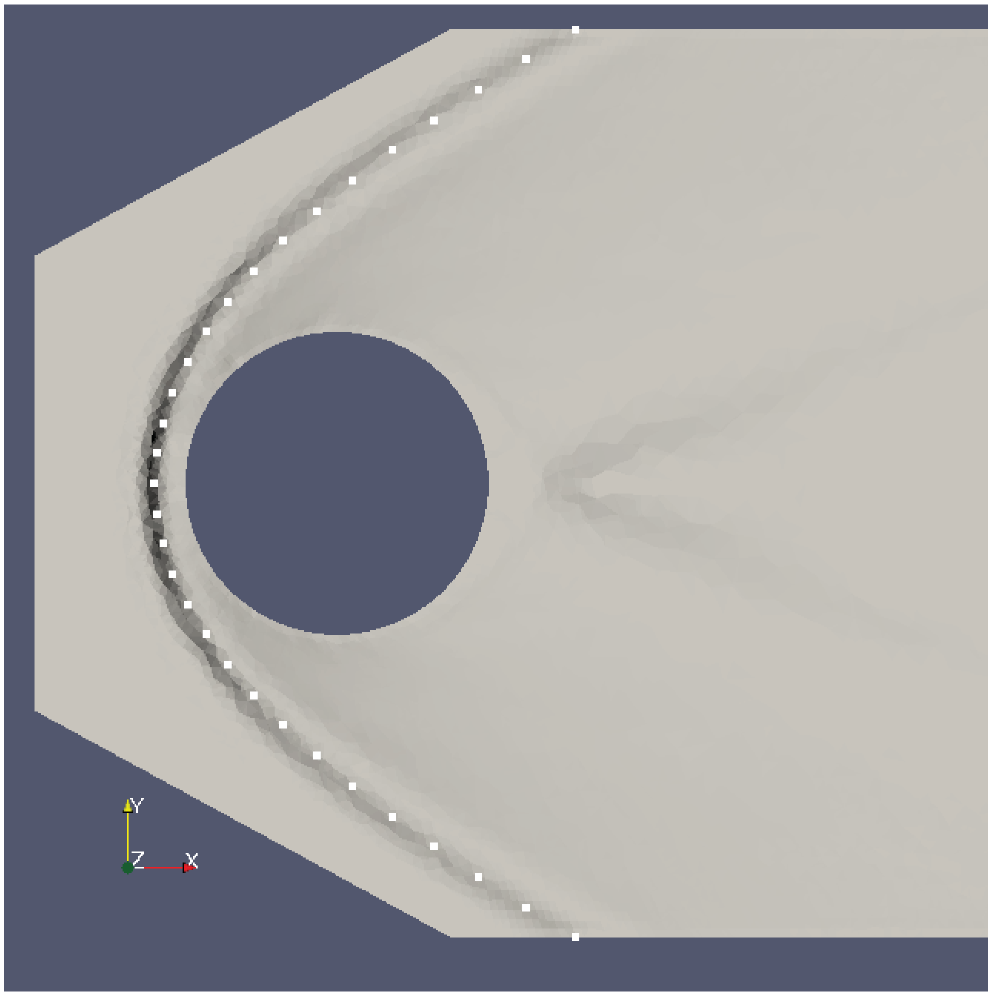


Figure 4.5: Convergence test of using fourth-order CESE code to solve a three-dimensional scalar advection equation.

figure shows the calculated shock profile generated by the new fourth-order CESE code compares well with the experimental relation shown in Eq. (4.42).

CHAPTER 5

NUMERICAL IMPLEMENTATION

The following section focus on the implementation of the various algorithms derived in Chapters 2-4. Previously the presentation of the these algorithms focused on generality and readability. In contrast, this chapter will focus on algorithms that are optimized for a particular set of physics or efficient algorithm deployment. This chapter will also provide an overview of SOLVCON[67], which is an unstructured CFD framework used during the course of the research.

5.1 One-Dimensional CESE Method

In Chapter 2 a generic method was used to relate the fluxes to the conserved variables, in this section another method is provided that uses the generalized Leibniz rule to express the derivatives of the product of two functions. This method was previously shown for the Euler equations by Dyson [79] and later by Dumbser and Munz [37].

To begin let $\mathcal{F}(x,t) = \mathcal{G}(x,t)\mathcal{H}(x,t)$ then any derivative of $\mathcal{F}$ may be expressed as

$$\frac{\partial^{n+m}\mathcal{F}}{\partial x^n \partial t^m} = \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\frac{\partial^{n+m-k-l}\mathcal{G}}{\partial x^{n-k}\partial t^{m-l}}\frac{\partial^{k+l}\mathcal{H}}{\partial x^k \partial t^l} \tag{5.1}$$

It should be noted that this procedure requires that all derivatives less than or equal to $n+m$ be known for $\mathcal{G}$ and $\mathcal{H}$.

It is possible to rewrite Eq. (5.1) to calculate any derivative of the quotient of two functions. To begin, take $\mathcal{G}(x,t) = \mathcal{F}(x,t)/\mathcal{H}(x,t)$ then any derivative of $\mathcal{G}$ may be calculated as.

$$\frac{\partial^{n+m}\mathcal{G}}{\partial x^n \partial t^m} = \frac{1}{h}\left[\frac{\partial^{n+m}\mathcal{F}}{\partial x^n \partial t^m} - \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\frac{\partial^{n+m-k-l}\mathcal{G}}{\partial x^{n-k}\partial t^{m-l}}\frac{\partial^{k+l}\mathcal{H}}{\partial x^k \partial t^l} \text{ for } (k,l) \neq (0,0)\right] \tag{5.2}$$

This equation requires that all derivatives less than or equal to $n+m$ of both $\mathcal{F}$ and $\mathcal{H}$ be known. The values $\mathcal{G}$ on the RHS of Eq. (5.2) only contain derivatives less than the one of the LHS. So as long as the derivatives of $\mathcal{G}$ are calculated in increasing order all values on the RHS are known. This means that no values of $\mathcal{G}$ needs to be known at the start of the algorithm.

These two equations are used to determine the flux and their spatial and temporal derivatives. To begin take the one-dimensional Euler equations

$$\frac{\partial}{\partial t}\begin{pmatrix}\rho \\ \rho v \\ \rho e\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}\rho v \\ \rho v^2 + p \\ (\rho e + p)v\end{pmatrix} = 0$$

Let u_1, u_2, u_3 represent the density, momentum and energy and f_1, f_2, f_3 be there respective fluxes. Then the Euler equations are rewritten as:

$$\frac{\partial}{\partial t}\begin{pmatrix}u_1 \\ u_2 \\ u_3\end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix}u_2 \\ (\gamma-1)u_3 + 1/2(3-\gamma)u_2^2/u_1 \\ \gamma u_2 u_3/u_1 - 1/2(\gamma-1)u_2^3/u_1^2\end{pmatrix} = 0 \tag{5.3}$$

Using Eq. (5.3) with Eq. (5.1) the fluxes and their derivatives are then expressed as a function of the conserved variables and their derivative.

$$\begin{aligned}
\frac{\partial^{n+m} f_1}{\partial x^n \partial t^m} &= \frac{\partial^{n+m} u_2}{\partial x^n \partial t^m} \\
\frac{\partial^{n+m} f_2}{\partial x^n \partial t^m} &= (\gamma-1)\frac{\partial^{n+m} u_3}{\partial x^n \partial t^m} + \frac{3-\gamma}{2}\frac{\partial^{n+m} u_2 v}{\partial x^n \partial t^m} \\
\frac{\partial^{n+m} f_3}{\partial x^n \partial t^m} &= \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\left(\gamma\frac{\partial^{n+m-k-l} u_3}{\partial x^{n-k}\partial t^{m-l}} - \frac{\gamma-1}{2}\frac{\partial^{n+m-k-l} u_2 v}{\partial x^{n-k}\partial t^{m-l}}\right)\frac{\partial^{k+l} v}{\partial x^k \partial t^l}
\end{aligned} \tag{5.4}$$

Velocity and kinetic energy are still present in Eq. (5.4) and are evaluated as:

$$\frac{\partial^{n+m} v}{\partial x^n \partial t^m} = \frac{\partial^{n+m}}{\partial x^n \partial t^m}\left(\frac{u_2}{u_1}\right) = \\ \frac{1}{u_1}\left[\frac{\partial^{n+m} u_2}{\partial x^n \partial t^m} - \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\frac{\partial^{n+m-k-l} v}{\partial x^{n-k}\partial t^{m-l}}\frac{\partial^{k+l} u_1}{\partial x^k \partial t^l} \text{ for } (k,l) \neq (0,0)\right] \tag{5.5}$$

and

$$\frac{\partial^{n+m} u_2 v}{\partial x^n \partial t^m} = \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\frac{\partial^{n+m-k-l} u_2}{\partial x^{n-k}\partial t^{m-l}}\frac{\partial^{k+l} v}{\partial x^k \partial t^l} \tag{5.6}$$

Since the LHS of the above equation is always the highest-order derivative of the unknown, everything on the RHS of the equation is known.

The procedure presented in Algorithm 1 is capable of calculating all of the required spatial and temporal derivatives of the conserved variables and fluxes. The only requirements are the conserved variables and their spatial derivatives. This procedure is inherently expandable to any desired order.

Algorithm 1 Procedure used to calculate the flux and temporal derivatives for the one-dimensional Euler Equations

```
$N \leftarrow$ Numerical order
for all $0 < m < N$ do
    for all $0 < n < N - m$ do
        $\frac{\partial^{n+m} v}{\partial x^n \partial t^m} \leftarrow \frac{1}{u_1}\left[\frac{\partial^{n+m} u_2}{\partial x^n \partial t^m} - \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\frac{\partial^{n+m-k-l} v}{\partial x^{n-k}\partial t^{m-l}}\frac{\partial^{k+l} u_1}{\partial x^k \partial t^l} \text{ for } (k,l) \neq (0,0)\right]$
        $\frac{\partial^{n+m} u_2 v}{\partial x^n \partial t^m} \leftarrow \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\frac{\partial^{n+m-k-l} u_2}{\partial x^{n-k}\partial t^{m-l}}\frac{\partial^{k+l} v}{\partial x^k \partial t^l}$
        $\frac{\partial^{n+m} f_1}{\partial x^n \partial t^m} \leftarrow \frac{\partial^{n+m} u_2}{\partial x^n \partial t^m}$
        $\frac{\partial^{n+m} f_2}{\partial x^n \partial t^m} \leftarrow (\gamma - 1)\frac{\partial^{n+m} u_3}{\partial x^n \partial t^m} + \frac{3-\gamma}{2}\frac{\partial^{n+m} u_2 v}{\partial x^n \partial t^m}$
        $\frac{\partial^{n+m} f_3}{\partial x^n \partial t^m} \leftarrow \sum_{k=0}^{n}\sum_{l=0}^{m}\binom{n}{k}\binom{m}{l}\left(\gamma\frac{\partial^{n+m-k-l} u_3}{\partial x^{n-k}\partial t^{m-l}} - \frac{\gamma-1}{2}\frac{\partial^{n+m-k-l} u_2 v}{\partial x^{n-k}\partial t^{m-l}}\right)\frac{\partial^{k+l} v}{\partial x^k \partial t^l}$
    end for
    if $m < N - 1$ then
        for all $0 < n \leq N - m - 1$ do
            $\frac{\partial^{n+m+1} u_1}{\partial x^n \partial t^{m+1}} \leftarrow -\frac{\partial^{n+1+m} f_1}{\partial x^{n+1} \partial t^m}$
            $\frac{\partial^{n+m+1} u_2}{\partial x^n \partial t^{m+1}} \leftarrow -\frac{\partial^{n+1+m} f_2}{\partial x^{n+1} \partial t^m}$
            $\frac{\partial^{n+m+1} u_3}{\partial x^n \partial t^{m+1}} \leftarrow -\frac{\partial^{n+1+m} f_3}{\partial x^{n+1} \partial t^m}$
        end for
    end if
end for
```

5.2 Multi-Dimensional Flux Calculation

The calculation of multi-dimensional fluxes presented in Chapters 3 and 4 are neither numerically efficient nor are they easy to extend to orders higher than fourth. Therefore, an approach, similar to the one used in Section 5.1, is applied to the multi-dimensional Euler Equations. What makes the two- and three-dimensional CESE methods more difficult than the one-dimensional CESE method is that the alternative rule for calculating the derivatives of the unknowns and the flux functions is inapplicable to the spatial derivatives. To proceed, let $\mathcal{F}$ be a function of three variables, $\mathcal{F}(\Psi_1, \Psi_2, \Psi_3) = \mathcal{G}(\Psi_1, \Psi_2, \Psi_3)\mathcal{H}(\Psi_1, \Psi_2, \Psi_3)$. Since the current research is limited to fourth-order in both two- and three-dimensions it is possible to simplify Eq. (5.1) and Eq. (5.2).

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \Psi_1} &= \sum_{i=0}^{1} \frac{\partial^{1-i}\mathcal{G}}{\partial \Psi_1^{1-i}} \frac{\partial^{i}\mathcal{H}}{\partial \Psi_1^{i}} \\
\frac{\partial^2 \mathcal{F}}{\partial \Psi_1 \partial \Psi_2} &= \sum_{i=0}^{1}\sum_{j=0}^{1} \frac{\partial^{2-i-j}\mathcal{G}}{\partial \Psi_1^{1-i}\Psi_2^{1-j}} \frac{\partial^{i+j}\mathcal{H}}{\partial \Psi_1^{i}\partial \Psi_2^{j}} \\
\frac{\partial^3 \mathcal{F}}{\partial \Psi_1 \partial \Psi_2 \partial \Psi_3} &= \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} \frac{\partial^{3-i-j-k}\mathcal{G}}{\partial \Psi_1^{1-i}\Psi_2^{1-j}\Psi_3^{1-k}} \frac{\partial^{i+j+k}\mathcal{H}}{\partial \Psi_1^{i}\partial \Psi_2^{j}\partial \Psi_3^{k}},
\end{aligned}$$

Where Ψ_1, Ψ_2, Ψ_3 may be any combination of $x,\ y,\ z$, and t. For easy reference let

$$\begin{aligned}
\mathcal{P}_{\Psi_1}(\mathcal{G}, \mathcal{H}) &\equiv \sum_{i=0}^{1} \frac{\partial^{1-i}\mathcal{G}}{\partial \Psi_1^{1-i}} \frac{\partial^{i}h}{\partial \Psi_1^{i}} \\
\mathcal{P}_{\Psi_1\Psi_2}(\mathcal{G}, \mathcal{H}) &\equiv \sum_{i=0}^{1}\sum_{j=0}^{1} \frac{\partial^{2-i-j}\mathcal{G}}{\partial \Psi_1^{1-i}\Psi_2^{1-j}} \frac{\partial^{i+j}\mathcal{H}}{\partial \Psi_1^{i}\partial \Psi_2^{j}} \\
\mathcal{P}_{\Psi_1\Psi_2\Psi_3}(\mathcal{G}, h) &\equiv \sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} \frac{\partial^{3-i-j-k}\mathcal{G}}{\partial \Psi_1^{1-i}\Psi_2^{1-j}\Psi_3^{1-k}} \frac{\partial^{i+j+k}\mathcal{H}}{\partial \Psi_1^{i}\partial \Psi_2^{j}\partial \Psi_3^{k}}.
\end{aligned}$$

These equation are easily modified to find the derivatives of the quotient of two functions.

$$\begin{aligned}
\mathcal{Q}_{\Psi_1}(\mathcal{G},\mathcal{H};\mathcal{F}) &\equiv \left(\frac{\partial \mathcal{F}}{\partial \Psi_1} - g\frac{\partial \mathcal{H}}{\partial \Psi_1}\right)\frac{1}{\mathcal{H}} \\
\mathcal{Q}_{\Psi_1\Psi_2}(\mathcal{G},\mathcal{H};\mathcal{F}) &\equiv \left(\frac{\partial^2 \mathcal{F}}{\partial \Psi_1 \partial \Psi_2} - \sum_{i=0}^{1}\sum_{i=0}^{1}\frac{\partial^{2-i-j}\mathcal{G}}{\partial \Psi_1^{1-i}\Psi_2^{1-j}}\frac{\partial^{i+j}\mathcal{H}}{\partial \Psi_1^{i}\partial \Psi_2^{j}}\right)\frac{1}{\mathcal{H}} \\
&\text{for } \big\{(i,j)\neq(0,0)\big\} \\
\mathcal{Q}_{\Psi_1\Psi_2\Psi_3}(\mathcal{G},\mathcal{H};\mathcal{F}) &\equiv \left(\frac{\partial^3 \mathcal{F}}{\partial \Psi_1 \partial \Psi_2 \partial \Psi_3} - \sum_{i=0}^{1}\sum_{i=0}^{1}\sum_{i=0}^{1}\frac{\partial^{3-i-j-k}\mathcal{G}}{\partial \Psi_1^{1-i}\Psi_2^{1-j}\Psi_3^{1-k}}\frac{\partial^{i+j+k}\mathcal{H}}{\partial \Psi_1^{i}\partial \Psi_2^{j}\partial \Psi_3^{k}}\right)\frac{1}{\mathcal{H}} \\
&\text{for } \big\{(i,j,k)\neq(0,0,0)\big\}
\end{aligned}$$

The Euler Equations may be written in the following generic form:

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \sum_{j=1}^{Ndim}\frac{\partial \rho v_j}{\partial x_j} = 0 \\
&\frac{\partial \rho v_i}{\partial t} + \sum_{j=1}^{Ndim}\rho v_i v_j + p\delta_{ij} = 0 \qquad \text{for } i = \{1,\ldots,Ndim\} \\
&\frac{\partial \rho e}{\partial t} + \sum_{j=1}^{Ndim}(\rho e + p)\, v_j = 0.
\end{aligned}$$

Pressure and velocity can be expressed in terms of the primary unknowns, or the conserved variables, i.e., u_i:

$$v_i = \frac{(\rho v_i)}{\rho}$$
$$p = (\gamma - 1)\left(\rho e - 0.5\rho \sum_{j}^{Ndim} \left(\frac{\rho v_j}{\rho}\right)^2\right)$$

The numerical procedure used to calculate the spatial and temporal derivatives of the fluxes and the temporal derivatives of the conserved variables uses a similar method as the one-dimensional method. For a two-dimensional Euler system let u_1, u_2, u_3, and u_4 represent the conserved variables ρ, ρv_x, ρv_y, and ρe and $f_1^{x,y}$, $f_2^{x,y}$, $f_3^{x,y}$, and $f_4^{x,y}$ be their respective fluxes. Since the order of differentiation for the spatial derivatives makes a difference the algorithm used to calculate the derivatives of the fluxes and conserved variables are more difficult. As such, instead of using a set of nested for loops, like Algorithm 1, a single temporal for loop is used and the spatial loop is replaced by a set of derivatives that changes based on the current temporal loop.

Algorithm 2 Procedure used to calculate the flux and temporal derivatives for the two-dimensional Euler Equations

```
$N \leftarrow 4$
for all $0 < m < N$ do
    for all $n$ in $\mathbf{\Psi}[m]$ do
        $\Psi_1 \leftarrow \Psi[m,n,1] \quad \Psi_2 \leftarrow \Psi[m,n,2] \quad \Psi_3 \leftarrow \Psi[m,n,3]$
        $\frac{\partial v_x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{Q}_{\mathbf{\Psi}[m,n]}(u_2, u_1; v_x)$
        $\frac{\partial v_y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{Q}_{\mathbf{\Psi}[m,n]}(u_3, u_1; v_y)$
        $\frac{\partial p}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow (\gamma - 1)\left(\frac{\partial u_4}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} - 0.5\mathcal{P}_{\mathbf{\Psi}[m,n]}(u_2, v_x) - 0.5\mathcal{P}_{\mathbf{\Psi}[m,n]}(u_3, v_y)\right)$
        $\frac{\partial f_1^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \frac{\partial u_2}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_1^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \frac{\partial u_3}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_2^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_2, v_x) + \frac{\partial p}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_2^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_2, v_y)$
        $\frac{\partial f_3^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_2, v_y)$
        $\frac{\partial f_3^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_3, v_y) + \frac{\partial p}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_4^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4 + P, v_x)$
        $\frac{\partial f_4^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4 + P, v_y)$
    end for
    if $m < N - 1$ then
        for all $0 < i \leq 4$ do
            for all $n$ in $\mathbf{\Psi}[m+1]$ do
                $\frac{\partial u_i}{\partial\mathbf{\Psi}[m+1,n]} \leftarrow -\frac{\partial}{\partial\mathbf{\Psi}[m+1,n]}\left(\frac{\partial f_i^x}{\partial x\partial t^{-1}} + \frac{\partial f_i^y}{\partial y\partial t^{-1}}\right)$
            end for
        end for
    end if
end for
```

The negative exponent in Algorithm 2 means that the temporal derivatives is reduced by one, e.g.

$$\frac{\partial}{\partial x\partial t}\left(\frac{\partial f_i^x}{\partial x\partial t^{-1}} + \frac{\partial f_i^x}{\partial y\partial t^{-1}}\right) \equiv \left(\frac{\partial f_i^x}{\partial x\partial x} + \frac{\partial f_i^y}{\partial y\partial x}\right)$$

In Algorithm 2 the values of $\boldsymbol{\Psi}$ are

derivatives with no temporal derivative

$$\boldsymbol{\Psi}[0,1] = \{0,0,0\} \quad \boldsymbol{\Psi}[0,2] = \{x,0,0\} \quad \boldsymbol{\Psi}[0,3] = \{y,0,0\}$$

$$\boldsymbol{\Psi}[0,4] = \{x,x,0\} \quad \boldsymbol{\Psi}[0,5] = \{x,x,x\} \quad \boldsymbol{\Psi}[0,6] = \{x,x,y\}$$

$$\boldsymbol{\Psi}[0,7] = \{x,y,0\} \quad \boldsymbol{\Psi}[0,8] = \{x,y,x\} \quad \boldsymbol{\Psi}[0,9] = \{x,y,y\}$$

$$\boldsymbol{\Psi}[0,10] = \{y,x,0\} \quad \boldsymbol{\Psi}[0,11] = \{y,x,x\} \quad \boldsymbol{\Psi}[0,12] = \{y,x,y\}$$

$$\boldsymbol{\Psi}[0,13] = \{y,y,0\} \quad \boldsymbol{\Psi}[0,14] = \{y,y,x\} \quad \boldsymbol{\Psi}[0,15] = \{y,y,y\}$$

Derivatives with one temporal derivative

$$\boldsymbol{\Psi}[1,1] = \{t,0,0\} \quad \boldsymbol{\Psi}[1,2] = \{x,t,0\} \quad \boldsymbol{\Psi}[1,3] = \{y,t,0\}$$

$$\boldsymbol{\Psi}[1,4] = \{x,x,t\} \quad \boldsymbol{\Psi}[1,5] = \{x,y,t\} \quad \boldsymbol{\Psi}[1,6] = \{y,x,t\}$$

$$\boldsymbol{\Psi}[1,7] = \{y,y,t\}$$

Derivatives with two temporal derivative

$$\boldsymbol{\Psi}[2,1] = \{t,t,0\} \quad \boldsymbol{\Psi}[2,2] = \{x,t,t\} \quad \boldsymbol{\Psi}[2,3] = \{y,t,t\}$$

Derivatives with three temporal derivative

$$\boldsymbol{\Psi}[3,1] = \{t,t,t\}$$

In Algorithm 2, $\mathcal{Q}_{\boldsymbol{\Psi}[m,n]} \equiv \mathcal{Q}_{\Psi[m,n,1]\Psi[m,n,2]\Psi[m,n,3]}$.

The same procedure is repeated for the three-dimensional Euler equations. In terms of methodology, The three-dimensional CESE method is nearly identical to the two-dimensional CESE method. One difference is that the number of unknowns that must be calculated is significantly larger. First, let u_1, u_2, u_3, u_4 and u_5 represent the conserved variables ρ, ρv_x, ρv_y, ρv_z, and ρe. Also let $f_1^{x,y,z}$, $f_2^{x,y,z}$, $f_3^{x,y,z}$, $f_4^{x,y,z}$, and $f_5^{x,y,z}$ be their respective fluxes.

Algorithm 3 Procedure used to calculate the flux and temporal derivatives for the three-dimensional Euler Equations

```
$N \leftarrow 4$
**for all** $0 < m < N$ **do**
    **for all** $n$ in $\mathbf{\Psi}[m]$ **do**
        $\Psi_1 \leftarrow \Psi[m,n,1] \quad \Psi_2 \leftarrow \Psi[m,n,2] \quad \Psi_3 \leftarrow \Psi[m,n,3]$
        $\frac{\partial v_x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{Q}_{\mathbf{\Psi}[m,n]}(u_2, u_1; v_x)$
        $\frac{\partial v_y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{Q}_{\mathbf{\Psi}[m,n]}(u_3, u_1; v_y)$
        $\frac{\partial v_z}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{Q}_{\mathbf{\Psi}[m,n]}(u_4, u_1; v_y)$
        $\frac{\partial p}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow (\gamma-1)\left(\frac{\partial u_4}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} - \sum_{j=0}^{Ndim} 0.5\mathcal{P}_{\mathbf{\Psi}[m,n]}(u_1, v_{x_j}^2)\right)$
        $\frac{\partial f_1^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \frac{\partial u_2}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_1^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \frac{\partial u_3}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_1^z}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \frac{\partial u_4}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_2^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_2, v_x) + \frac{\partial p}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_2^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_2, v_y)$
        $\frac{\partial f_2^z}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_2, v_z)$
        $\frac{\partial f_3^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_3, v_x)$
        $\frac{\partial f_3^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_3, v_y) + \frac{\partial p}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_3^z}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_3, v_z)$
        $\frac{\partial f_4^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4, v_x)$
        $\frac{\partial f_4^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4, v_y)$
        $\frac{\partial f_4^z}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4, v_z) + \frac{\partial p}{\partial\Psi_1\partial\Psi_2\partial\Psi_3}$
        $\frac{\partial f_5^x}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4 + P, v_x)$
        $\frac{\partial f_5^y}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4 + P, v_y)$
        $\frac{\partial f_5^z}{\partial\Psi_1\partial\Psi_2\partial\Psi_3} \leftarrow \mathcal{P}_{\mathbf{\Psi}[m,n]}(u_4 + P, v_z)$
    **end for**
(continued)
```

Algorithm 3 Procedure for calculating fluxes and their temporal derivatives of the three-dimensional Euler Equations (continued)

```
        if $m < N - 1$ then
            for all $0 < i \leq 4$ do
                for all $n$ in $\mathbf{\Psi}[m+1]$ do
                    $\frac{\partial u_i}{\partial \mathbf{\Psi}[m+1,n]} \leftarrow -\frac{\partial}{\partial \mathbf{\Psi}[m+1,n]} \left( \frac{\partial f_i^x}{\partial x \partial t^{-1}} + \frac{\partial f_i^y}{\partial y \partial t^{-1}} + \frac{\partial f_i^z}{\partial z \partial t^{-1}} \right)$
                end for
            end for
        end if
    end for
```

In Algorithm 3 the values of $\mathbf{\Psi}$ are

derivatives with no temporal derivative

$\mathbf{\Psi}[0,1] = \{0,0,0\}$	$\mathbf{\Psi}[0,2] = \{x,0,0\}$	$\mathbf{\Psi}[0,3] = \{y,0,0\}$	$\mathbf{\Psi}[0,4] = \{z,0,0\}$
$\mathbf{\Psi}[0,5] = \{x,x,0\}$	$\mathbf{\Psi}[0,6] = \{x,x,x\}$	$\mathbf{\Psi}[0,7] = \{x,x,y\}$	$\mathbf{\Psi}[0,8] = \{x,x,z\}$
$\mathbf{\Psi}[0,9] = \{x,y,0\}$	$\mathbf{\Psi}[0,10] = \{x,y,x\}$	$\mathbf{\Psi}[0,11] = \{x,y,y\}$	$\mathbf{\Psi}[0,12] = \{x,y,z\}$
$\mathbf{\Psi}[0,13] = \{x,z,0\}$	$\mathbf{\Psi}[0,14] = \{x,z,x\}$	$\mathbf{\Psi}[0,15] = \{x,z,y\}$	$\mathbf{\Psi}[0,16] = \{x,z,z\}$
$\mathbf{\Psi}[0,17] = \{y,x,0\}$	$\mathbf{\Psi}[0,18] = \{y,x,x\}$	$\mathbf{\Psi}[0,19] = \{y,x,y\}$	$\mathbf{\Psi}[0,20] = \{y,x,z\}$
$\mathbf{\Psi}[0,21] = \{y,y,0\}$	$\mathbf{\Psi}[0,22] = \{y,y,x\}$	$\mathbf{\Psi}[0,23] = \{y,y,y\}$	$\mathbf{\Psi}[0,24] = \{y,y,z\}$
$\mathbf{\Psi}[0,25] = \{y,z,0\}$	$\mathbf{\Psi}[0,26] = \{y,z,x\}$	$\mathbf{\Psi}[0,27] = \{y,z,y\}$	$\mathbf{\Psi}[0,28] = \{y,z,z\}$
$\mathbf{\Psi}[0,29] = \{z,x,0\}$	$\mathbf{\Psi}[0,30] = \{z,x,x\}$	$\mathbf{\Psi}[0,31] = \{z,x,y\}$	$\mathbf{\Psi}[0,32] = \{z,x,z\}$
$\mathbf{\Psi}[0,33] = \{z,y,0\}$	$\mathbf{\Psi}[0,34] = \{z,y,x\}$	$\mathbf{\Psi}[0,35] = \{z,y,y\}$	$\mathbf{\Psi}[0,36] = \{z,y,z\}$
$\mathbf{\Psi}[0,37] = \{z,z,0\}$	$\mathbf{\Psi}[0,38] = \{z,z,x\}$	$\mathbf{\Psi}[0,39] = \{z,z,y\}$	$\mathbf{\Psi}[0,40] = \{z,z,z\}$

derivatives with one temporal derivative

$$\mathbf{\Psi}[1,1] = \{t,0,0\} \quad \mathbf{\Psi}[1,2] = \{x,t,0\} \quad \mathbf{\Psi}[1,3] = \{y,t,0\} \quad \mathbf{\Psi}[1,4] = \{z,t,0\}$$

$$\mathbf{\Psi}[1,5] = \{x,x,t\} \quad \mathbf{\Psi}[1,6] = \{x,y,t\} \quad \mathbf{\Psi}[1,7] = \{x,z,t\}$$

$$\mathbf{\Psi}[1,8] = \{y,x,t\} \quad \mathbf{\Psi}[1,9] = \{y,y,t\} \quad \mathbf{\Psi}[1,10] = \{y,z,t\}$$

$$\mathbf{\Psi}[1,11] = \{z,x,t\} \quad \mathbf{\Psi}[1,12] = \{z,y,t\} \quad \mathbf{\Psi}[1,13] = \{z,z,t\}$$

derivatives with two temporal derivative

$$\mathbf{\Psi}[2,1] = \{t,t,0\} \quad \mathbf{\Psi}[2,2] = \{x,t,t\} \quad \mathbf{\Psi}[2,3] = \{y,t,t\} \quad \mathbf{\Psi}[2,4] = \{z,t,t\}$$

derivatives with three temporal derivative

$$\mathbf{\Psi}[3,1] = \{t,t,t\}$$

5.3 SOLVCON: Numerical Framework

This section details how the various equations derived in Chapters 3 and 4 are deployed in the numerical framework. During the development of the multidimensional CESE code the CFD framework, SOLVCON, was chosen to aid in the development effort. By using the framework provided by SOLVCON more time could be dedicated to implementing the new algorithms rather than basic code development. Such basic code development includes features common to all contemporary high performance CFD codes designed to run in a parallel environment such as: I/O, domain decomposition, and message passing.

SOLVCON was developed in house at the OSU CFD lab by Chen[67] and was written in a combination of Python and C. Python provides the object orientated aspect of the framework and the C code is used to implement the numerical algorithms. In other words Python provides the logic and call structure necessary for the program to run while the numerically expensive routines are written in C for efficiency. SOLVCON was originally developed as a second order unstructured CESE solver. Since the second- and fourth-order versions of the CESE algorithms are very similar, only the part of the codes related to the space-time integration and the data structure had to be modified for the implementation of the fourth-order CESE method.

The majority of the algorithms used internally by SOLVCON for evaluating the fluxes across the bottom/top and side surfaces of a typical CE are the same for two- and three-dimensional CESE methods. This similarity greatly simplifies the work required to maintain the code. The difference between the two- and three-dimensional procedures can be easily handled by having the variable length loops or through pre-compiler macros. The resulting treatment compiles two separate libraries from the same source code. One library for the two-dimensional solver and another for the three-dimensional solver.

Although the same numerical algorithm is used for both two- and three-dimensional problems, the procedure is more readable when the algorithm for two- and three-dimensional problems are separated.

Algorithm 4 Algorithm for calculating flux through the side surface for the two-dimensional CESE method.

```
for all Cells in Mesh do
    $icl \leftarrow$ current cell
    $F_{side}[icl] \leftarrow 0;$
    for all Adjacent Cells of $icl$ do
        $jcl \leftarrow$ current adjacent cell
        $\mathbf{x}_j \leftarrow SolutionPointCoordinate(jcl)$
        $iface \leftarrow$ shared face
        $\mathbf{x}_2 \leftarrow CellCenterCoordinate(jcl)$
        $f^{*x^i}_{jcl} \leftarrow$ Flux calculated at adjacent cell, $jcl$
        for all nodes on $iface$ do
            $\mathbf{x}_1 \leftarrow NodeCoordinates$
            $F_{side}[icl]$ += [Eq. (3.24)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_j, \Delta t, f^{x^i}_{jcl})$
        end for
    end for
end for
```

where $F_{side}[icl]$ is the sum of the flux through all side faces for cell icl and $f^{*x^i}_{jcl}$ contains all derivatives of the flux.

Algorithm 5 Algorithm for calculating flux through bottom surface for the two-dimensional CESE scheme

```
for all Cells in Mesh do
    $icl \leftarrow$ current cell
    for all Adjacent Cells of $icl$ do
        $jcl \leftarrow$ current adjacent cell
        $iface \leftarrow$ shared face
        $u^*_{jcl} \leftarrow$ solution at $jcl$ from previous time step
        $F_{bottom}(icl, iface) \leftarrow 0$
        $\mathbf{x}_1 \leftarrow CellCenterCoordinate(jcl)$
        $\mathbf{x}_2 \leftarrow NodeCoordinates(first\ node\ on\ face)$
        $\mathbf{x}_4 \leftarrow NodeCoordinates(second\ node\ on\ face)$
        $\mathbf{x}_3 \leftarrow CellCenterCoordinate(icl)$
        $\mathbf{x}_j \leftarrow SolutionPointCoordinate(jcl)$
        $F_{bottom}(icl, iface)$ += [Eq. (3.30)] $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_j, u_{jcl})$
        $F_{bottom}(icl, iface)$ += [Eq. (3.30)] $(\mathbf{x}_1, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_j, u_{jcl})$
    end for
end for
```

where $F_{bottom}(icl, iface)$ is the sum of the flux through all bottom faces for cell icl and u^*_{jcl} contains all derivatives of the conserved variables.

Algorithm 6 Algorithm for calculating flux through top surface for the two-dimensional CESE scheme

```
for all Cells in Mesh do
    $icl \leftarrow$ current cell
    $F_{top}[icl] \leftarrow 0$
    $u^*_{icl} \leftarrow$ solution at $icl$ from current time step
    $\mathbf{x}_j \leftarrow SolutionPointCoordinate(icl)$
    for all Adjacent Cells of $icl$ do
        $jcl \leftarrow$ current adjacent cell
        $iface \leftarrow$ shared face
        $inode \leftarrow$ current node
        $\mathbf{x}_1 \leftarrow CellCenterCoordinate(icl)$
        $\mathbf{x}_2 \leftarrow NodeCoordinates(first\ node\ on\ face)$
        $\mathbf{x}_3 \leftarrow CellCenterCoordinate(jcl)$
        $\mathbf{x}_4 \leftarrow NodeCoordinates(second\ node\ on\ face)$
        $F_{top}[icl]$ += [Eq. (3.33)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_j, u_{icl})$
        $F_{top}[icl]$ += [Eq. (3.33)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_j, u_{icl})$
    end for
end for
```

where $F_{top}[icl]$ is the sum of the flux through all top faces for cell icl and u^*_{icl} contains all derivatives of the conserved variables.

Algorithm 7 Algorithm for calculating flux through the side surface for the three-dimensional CESE scheme

```
for all Cells in Mesh do
    $icl \leftarrow$ current cell
    $F_{side}[icl] \leftarrow 0;$
    for all Adjacent Cells of $icl$ do
        $jcl \leftarrow$ current adjacent cell
        $\mathbf{x}_j \leftarrow SolutionPointCoordinate(jcl)$
        $\mathbf{x}_3 \leftarrow CellCenterCoordinate(jcl)$
        $iface \leftarrow$ shared face
        $f^{*x^i}_{jcl} \leftarrow$ Flux calculated at adjacent cell, $jcl$
        for all edges on $iface$ do
            $\mathbf{x}_1 \leftarrow NodeCoordinates(first\ node\ on\ edge)$
            $\mathbf{x}_2 \leftarrow NodeCoordinates(second\ node\ on\ edge)$
            $F_{side}[icl]$ += [Eq. (4.15)] $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_j, \Delta t, f^{x^i}_{jcl})$
        end for
    end for
end for
```

where $F_{side}[icl]$ is the sum of the flux through all side faces for cell icl and $f^{*x^i}_{jcl}$ contains all derivatives of the flux.

Algorithm 8 Algorithm for calculating flux through the bottom surface of a CE in the three-dimensional CESE method.

```
for all Cells in Mesh do
    $icl \leftarrow$ current cell
    for all Adjacent Cells of $icl$ do
        $jcl \leftarrow$ current adjacent cell
        $iface \leftarrow$ shared face
        $u^*_{jcl} \leftarrow$ solution at $jcl$ from previous time step
        $F_{bottom}(icl, iface) \leftarrow 0$
        for all edges on $iface$ do
            $\mathbf{x}_1 \leftarrow FaceCenterCoordinates(iface)$
            $\mathbf{x}_2 \leftarrow NodeCoordinates(first\ node\ on\ edge)$
            $\mathbf{x}_3 \leftarrow NodeCoordinates(second\ node\ on\ edge)$
            $\mathbf{x}_4 \leftarrow CellCenterCoordinate(jcl)$
            $\mathbf{x}_j \leftarrow SolutionPointCoordinate(jcl)$
            $F_{bottom}(icl, iface)$ += [Eq. (4.19)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_j, u_{jcl})$
            $\mathbf{x}_4 \leftarrow CellCenterCoordinate(icl)$
            $F_{bottom}(icl, iface)$ += [Eq. (4.19)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_j, u_{jcl})$
        end for
    end for
end for
```

where $F_{bottom}(icl, iface)$ is the sum of the flux through all bottom faces for cell icl and u^*_{jcl} contains all derivatives of the conserved variables.

Algorithm 9 Algorithm for calculating flux through top surface for the three-dimensional CESE scheme

```
for all Cells in Mesh do
    $icl \leftarrow$ current cell
    $F_{top}[icl] \leftarrow 0$
    $u^*_{icl} \leftarrow$ solution at $icl$ from current time step
    $\mathbf{x}_j \leftarrow SolutionPointCoordinate(icl)$
    for all Adjacent Cells of $icl$ do
        $jcl \leftarrow$ current adjacent cell
        $iface \leftarrow$ shared face
        for all edges on $iface$ do
            $inode \leftarrow$ current node
            $\mathbf{x}_1 \leftarrow FaceCenterCoordinates(iface)$
            $\mathbf{x}_2 \leftarrow NodeCoordinates(first\ node\ on\ edge)$
            $\mathbf{x}_3 \leftarrow NodeCoordinates(second\ node\ on\ edge)$
            $\mathbf{x}_4 \leftarrow CellCenterCoordinate(jcl)$
            $F_{top}[icl]$ += [Eq. (4.20)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_j, u_{icl})$
            $\mathbf{x}_4 \leftarrow CellCenterCoordinate(icl)$
            $F_{top}[icl]$ += [Eq. (4.20)]$(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_j, u_{icl})$
        end for
    end for
end for
```

where $F_{top}[icl]$ is the sum of the flux through all top faces for cell icl and u^*_{icl} contains all derivatives of the conserved variables.

5.4 Call Structure of the Fourth-Order CESE Method

The fourth-order numerical procedure detailed in Section 3.5 and Section 4.5 are not an efficient use of computational resources. To make an efficient computational algorithm requires the determination of which variables to store for future use and which ones will be recalculate as needed. This decision is highly dependent on the computational architecture that the code is running on. For example on a vector machine, such as GPUs, the amount of resources dedicated to floating point operations is much greater than that for fetching memory, so storing variables for future use may not be as efficient as calculating them on the fly. On the other hand, the CPU is much more efficient at retrieving memory then the GPU so storing variables for future use may be more efficient. The problem faced in development of the numerical algorithms was that a large amount of memory needs to be stored in each cell.

A fourth-order CESE scheme requires the storage of 30 doubles for the two-dimensional CESE algorithm and 80 doubles for the three-dimensional algorithm. The storage requirements takes into account the requirement that the primary variables needs to be stored at two different time steps. Compare this with the second-order scheme which requires 6 and 8 doubles per governing equation for the two- and three-dimensional schemes respectively. Ideally there are several other variables associated with the CESE scheme that could be cached for future use. For example the temporal derivatives of the conserved variables, the fluxes and their spatial and temporal derivatives, as well as the integration over the bottom, top and side surfaces.

The cost associated with caching each of the variables is sown in Table 5.1. Since the memory requirement of the mesh is relatively small they are not included in this table.

Table 5.1: Table of variables and their storage requirements for a fourth-order CESE scheme

	2D	3D	
Primary Variables	30	80	per governing equation per cell
Temporal Derivatives	11	18	per governing equation per cell
Fluxes all derivatives	52	164	per governing equation per cell
Top surface integrals	10	20	per cell
Bottom surface integrals	10	20	per neighboring cell per cell
Side surface integrals	20	60	per neighboring cell per cell

Based upon how often the values are used, how often they change and how expensive they are to calculate the only additional variables that are cached for future use are the temporal derivatives of the primary variables and the top/bottom integrations results. Even though the side integration does not change during the simulation it requires quite a bit of memory and are relatively quick integrations. The motivation behind caching the temporal derivatives is that they are used in all of the routines during each temporal step. The fluxes and their derivatives requires the most amount of memory to store and, depending on the governing equation, could potentially be the most expensive variables to calculate. As such a scheme that minimizes the number of times they need to be recalculated is paramount to an efficient code. Algorithm 10 presents a more numerically efficient procedure than the ones outline in Section 3.5 and Section 4.5. This method minimizing the number of times the flux needs to be recalculated.

Algorithm 10 Outline of the numerical procedure.

```
for all Time in Simulation do
    for all Cells in Mesh do
        $icl \leftarrow$ current cell
        $solt[icl] \leftarrow$ Calculate the temporal derivatives of the primary unknowns $u_i$
    end for
    for all Cells in Mesh do
        $icl \leftarrow$ current cell
        for all Neighboring cells do
            $jcl \leftarrow$ adjacent cells
            $solt \leftarrow solt[jcl]$
            $F \leftarrow$ Calculate the fluxes and their derivatives
            Calculate the second derivatives of $u_i$ by space-time integration of the additional equations
            Bottom and side surface space-time integration for $u_i$
        end for
    end for
    for all Cells in Mesh do
        Calculate the third derivatives of $u_i$
    end for
    for all Cells in Mesh do
        Top surface space-time procedure for calculating the conserved variables $u_i$
    end for
    for all Cells in Mesh do
        Calculate the first derivatives of $u_i$
    end for
end for
```

In Algorithm 10, the use of brackets [ ] signifies that a variable is either being stored or retrieved. The key difference between this procedure and the ones outlined in Section 3.5 and Section 4.5 is that the space-time integration is divided into parts that are either dependent on the solutions at the previous time step or the solutions to be solved at the current time step. The values of the flux functions are readily calculated once the primary unknowns are known. Thus, the operation count of calculating the fluxes is reduced by almost half. By using this procedure, the fluxes need to be calculated once during the calculation of the temporal derivatives and again N_{face} times during the space-time integration over the surfaces of the cell. Here N_{face} is the number of surfaces of the cell. In two-dimensional cases with the use of a triangular mesh, the flux function will be calculated four times.

It is possible to reduce the operational count related to the flux calculation. In the optimum condition, the flux could be calculated only once. However, this requires very careful programing to avoid the so called race condition. A race condition occurs when two or more computational threads attempt to modify the value of the same variable at the same time. The error due to the race condition would not occur if the program was running in a serial mode, or if each thread had its own memory when the program runs using multiple threads. However, SOLVCON is designed to run in a parallel mode that uses both shared and distributed memory.

The numerical algorithms put forth in this chapter are meant to provide a detailed outline of the numerical implementation of equation that were derived in Chapters 2, 3, and 4. Some algorithms presented the equation exactly as they were presented while others provided a more numerically efficient approach to achieve the same results. By using the equations derived in the earlier chapters and the algorithms presented in this chapter future researchers should be able to use these as a template in creation of their own high-order CESE code.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 Achievements

In this dissertation, the original second-order space-time CESE Method has been extended to a hierarchy of high-order CESE methods. The newly developed algorithms are general and are valid for all high-order of accuracy for the CESE methods, including the fourth-, sixth-, eighth-order, etc. The development of the new algorithms include a hierarchy of space-time integration equations for time-marching solutions in one-, two- and three-dimensional spaces. For two- and three-dimensional problems, unstructured meshes are employed for spatial discretization. Numerical implementation of the new high-order methods include one-, two-, and three-dimensional computer codes for time-accurate numerical solutions of the Euler equations for compressible flows. For two- and three-dimensional codes, the new high-order CESE codes are developed based on the use of unstructured meshes with capabilities of using mixed elements. For two-dimensional simulations, unstructured meshes could be composed of a mixture of triangles and quadrilaterals. For three-dimensional problems, the mesh employed could be composed of tetrahedrons, pyramids, prisms, and hexahedrons.

The new high-order CESE methods retains the following favorable characteristics of the original second-order CESE method: (i) The method is based on integrating the unique space-time integral equation for the time marching calculation. (ii) The mesh stencil is compact in the sense that only the cell adjoint to the central cell where the solution at the new time level is sought are involved in the algorithm. (iii) For stable calculation, the CFL number must be less than 1. This CFL constraint is identical to that of the original second-order CESE method. (iv) The algorithms are truly multi-dimensional in the sense that no directional splitting is used in the algorithm. (v) The algorithms as well as the associated codes can handle complex geometries by using unstructured meshes. The codes were written so that mesh composed of mixed types of cells can be used.

In this development, the one-, two-, and three-dimensional Euler equations for compressible flows were used as a model equation. The new high-order CESE codes have been developed in a computational framework, SOLVCON. Therefore, the high-order CESE solvers can be readily applied to solve linear or nonlinear hyperbolic PDEs as well as conservation laws other than the Euler equations.

The newly developed one-, two-, and three-dimensional high-order CESE methods as well as the associated codes have been extensively tested by numerical simulations of well known benchmarked problems defined in one-, two- and three-dimensional space. The numerical results show that the new high-order CESE methods and codes can achieve the desired order of accuracy and can accurately resolve solution discontinuities incurred by shock waves in supersonic compressible flows.

The new high-order CESE solvers have been successfully developed and validated. The resultant methods and codes fill the void in the existing capabilities of the second-order CESE method. More than all, the results of the present work also demonstrate a new direction in developing high-order, unstructured-meshed CFD methods and codes.

By using the one-dimensional CESE method and codes, numerical solutions of linear wave equations up to twentieth-order in accuracy have been reported. For solving the nonlinear Euler equations, results up to twelfth-order are reported. For two- and three-dimensional problems, fourth-order methods and codes have been developed. The methods and codes were verified and validate for solving the linear advection equation as well as the nonlinear Euler equations. Theoretically, the two- and three-dimensional, fourth-order solvers can be straightforwardly extended to be sixth-, eight-, and higher-order. However, these multi-dimensional solvers were not developed due to the memory requirements and associated code complexity.

6.2 Future Research

The newly developed, high-order, multi-dimensional solvers open the door for the following new areas of algorithm and code development: (i) The solving of other flow physics such as the Navier-Stokes equations for viscous flows. (ii) The development of a hybrid second/fourth-order solver. (iii) The optimizing the weighing schemes for the higher-order methods. (iv) The development of higher-order boundary condition treatments. (v) The effect that the alternate rule of differentiation has on the accuracy of the solution.

In what follows, further illustration is provided for each proposed task.

Some of these topics are already being addressed by other researchers. For example, Chang[63] has developed a Navier-Stokes solver that employs the alternative rule for differentiation. By applying the alternate rule, the number of conserved variables is reduced considerably. For two-dimensions, the number of the unknowns drops from 15 to 10. For three-dimensional problems, the number of unknowns drops from 40 to 20. The savings will be even more substantial for sixth- and higher-order solvers. Not only the required memory goes down but also the internal algorithms become simpler. The development of a high-order, Navier-Stokes solver is also important. When using the second-order CESE method, a first-order Taylor series is employed. However, the viscous terms are second-order derivatives, and there is no clear method to calculate the viscous terms. For the fourth-order CESE method, the third-order Taylor series are used to discretize the primary unknowns and flux functions. As such, the second-order derivatives of the primary unknowns can be well defined and be used in calculating the viscous terms.

A hybrid second/fourth-order solver could be more computationally efficient when compared to the fourth-order CESE solver. Essentially, in such a hybrid solver, the second-order CESE method and the fourth-order CESE method are used simultaneously in the same simulation. One possibility for a scheme like this would be to apply a second-order solver to the free-stream region of the compressible flow being simulated, where the flow is essentially steady and uniform and thus the use of the original second-order method could significantly reduce the operational counts of the overall computational task.

Another possibility for a hybrid code is to apply the second-order CESE method in regions where a strong shock is present [63]. Essentially, a shock sensor could be used to detect the presence of shock waves. When a shock is detected, the numerical scheme employed would be automatically changed to be the second-order CESE method, which in turn provides a suitable platform for using the limiter to capture the shock waves.

In all results presented in the present dissertation, the second-order boundary condition has been applied. For example, a slip boundary condition parallel to the wall was applied as the wall condition. At the far field and outlets, the non-reflective boundary conditions have been applied. In such boundary condition treatments, the second- and third-order derivatives were set to be null. A more accurate boundary condition treatment could be possible, if the second and third derivatives were computed according to certain logics. Another approach to the higher-order boundary condition treatments is to consider the curvature of the wall. One possible approach to find the derivatives would be implementing a flux conservative boundary condition. Although previous attempts have shown little benefit from a flux conservative boundary condition treatment, the addition of higher-order terms may change this previous finding.

Another possible topic for future research would be to optimize the reweighing scheme to be higher orders. One potential improvement would be utilizing the multiple representations of the same derivatives during the reweighing calculation. For example, consider the following derivatives $\partial^3 u/\partial x\partial x\partial y$, $\partial^3 u/\partial x\partial y\partial x$ and $\partial^3 u/\partial y\partial x\partial x$. All above three derivatives represent the same change in the conserved quantity u but are calculated using three different governing equations with different primary unknowns, i.e., the additional equations for $\partial^2 u/\partial x\partial x$, $\partial^2 u/\partial x\partial y$ and $\partial^2 u/\partial y\partial x$. It is currently unknown if these three governing equations would have the same stability characteristics when a solution discontinuity is present in the flow filed. It is possible to use the reweighing idea, if the alternative differentiating rule is applicable. In such case, only the third additional governing equation for $\frac{\partial^2 u}{\partial y\partial x}$ could be dropped due to the alternative rule assumption.

BIBLIOGRAPHY

[1] B. Cockburn, C.-W. Shu, Runge–Kutta discontinuous galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (3) (2001) 173–261.

[2] Y. Liu, M. Vinokur, Z. J. Wang, Spectral (finite) volume method for conservation laws on unstructured grids v: Extension to three-dimensional systems, Journal of Computational Physics 212 (2) (2006) 454–472.

[3] S.-C. Chang, A new approach for constructing highly stable high order CESE schemes, in: 48th AIAA Aerospace Science Meeting, Vol. 2010-543, AIAA, AIAA-2010-543, 48th AIAA Aerospace Science Meeting, Orlando, FL, 2010.

[4] S.-C. Chang, W.-M. To, A new numerical framework for solving conservation laws: The method of space-time conservation element and solution element, Tech. Rep. E-6403; NAS 1.15:104495; NASA-TM-104495 (Aug. 1991).

[5] S.-C. Chang, The method of space-time conservation element and solution element – a new approach for solving the navier-stokes and euler equations, Journal of Computational Physics 119 (2) (1995) 295–324.

[6] Y. Liu, M. Vinokur, Z. Wang, Spectral difference method for unstructured grids i: Basic formulation, Journal of Computational Physics 216 (2) (2006) 780–801.

[7] H. Luo, J. D. Baum, R. Löhner, A p-multigrid discontinuous galerkin method for the euler equations on unstructured grids, Journal of Computational Physics 211 (2) (2006) 767–783.

[8] T. J. Barth, P. O. Frederickson, Higher order solution of the euler equations on unstructured grids using quadratic reconstruction, in: 28th AIAA Aeerospace Sciences Meeting, Vol. 1, Reno, NV, 1990.

[9] T. J. Barth, Recent developments in high order k-exact reconstruction on unstructured meshes, in: 31st AIAA Aerospace Science Meeting and Exhibit, AIAA, Reno, NV, 1993.

[10] Z. J. Wang, Y. Liu, Spectral (finite) volume method for conservation laws on unstructured grids, Journal of Computational Physics 179 (2) (2002) 665–697.

[11] B. Cockburn, S.-Y. Lin, C.-W. Shu, TVB runge-kutta local projection discontinuous galerkin finite element method for conservation laws III: one-dimensional systems, Journal of Computational Physics 84 (1) (1989) 90–113.

[12] B. Cockburn, C.-W. Shu, TVB runge-kutta local projection discontinuous galerkin finite element method for conservation laws II: general framework, Mathematics of Computation 52 (186) (1989) 411–435.

[13] B. Cockburn, S. Hou, C.-W. Shu, The runge-kutta local projection discontinuous galerkin finite element method for conservation laws. IV: the multidimensional case, Mathematics of Computation 54 (190) (1990) 545–581.

[14] L. J. Durlofsky, B. Engquist, S. Osher, Triangle based adaptive stencils for the solution of hyperbolic conservation laws, Journal of Computational Physics 98 (1) (1992) 64–73.

[15] R. Abgrall, On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation, Journal of Computational Physics 114 (1) (1994) 45 – 58.

[16] C. F. Ollivier-Gooch, Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction, Journal of Computational Physics 133 (1) (1997) 6–17.

[17] O. Friedrich, Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids, Journal of Computational Physics 144 (1) (1998) 194–212.

[18] M. Dumbser, M. Käser, Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems, Journal of Computational Physics 221 (2) (2007) 693–723.

[19] M. Dumbser, M. Käser, V. A. Titarev, E. F. Toro, Quadrature-free non-oscillatory finite volume schemes on unstructured meshes for nonlinear hyperbolic systems, Journal of Computational Physics 226 (1) (2007) 204–243.

[20] E. F. Toro, R. Millington, A. S. Nejad, Towards very high order godunov schemes, in: Godunov Methods, Theory and Applications, Kluwer/Plenum Academic Publishers, 2001, pp. 905–935.

[21] Z. J. Wang, Spectral (finite) volume method for conservation laws on unstructured grids. basic formulation, Journal of Computational Physics 178 (1) (2002) 210–251.

[22] M. Delanaye, Y. Liu, Quadratic reconstruction finite volume schemes on 3D arbitrary unstructured polyhedral grids, in: AIAA Computational Fluid Dynamics Conference, 14 th, Norfolk, VA, Vol. AIAA-1999-3259, AIAA, Norfolk, VA, 1999.

[23] K. Van den Abeele, T. Broeckhoven, C. Lacor, Dispersion and dissipation properties of the 1D spectral volume method and application to a p-multigrid algorithm, Journal of Computational Physics 224 (2) (2007) 616–636.

[24] K. Van den Abeele, C. Lacor, An accuracy and stability study of the 2D spectral volume method, Journal of Computational Physics 226 (1) (2007) 1007–1026.

[25] K. Van den Abeele, G. Ghorbaniasl, M. Parsani, C. Lacor, A stability analysis for the spectral volume method on tetrahedral grids, Journal of Computational Physics 228 (2) (2009) 257–265.

[26] R. Harris, Z. Wang, Y. Liu, Efficient quadrature-free high-order spectral volume method on unstructured grids: Theory and 2D implementation, Journal of Computational Physics 227 (3) (2008) 1620–1642.

[27] M. Yang, R. Harris, Z. Wang, Y. Liu, Efficient quadrature-free 3D high-order spectral volume method on unstructured grids, in: 18th AIAA Computational Fluid Dynamics Conference, Vol. 2007-4325, AIAA, Miami, Fl, 2007.

[28] C. Breviglieri, J. L. F. Azevedo, E. Basso, M. A. F. Souza, Implicit high-order spectral finite volume method for inviscid compressible flows, AIAA Journal 48 (10) (2010) 2365–2376.

[29] Z. J. Wang, Y. Liu, G. May, A. Jameson, Spectral difference method for unstructured grids II: extension to the euler equations, Journal of Scientific Computing 32 (1) (2006) 45–71.

[30] K. Van den Abeele, C. Lacor, Z. J. Wang, On the stability and accuracy of the spectral difference method, Journal of Scientific Computing 37 (2) (2008) 162–188.

[31] W. H. Reed, T. R. Hill, TRIANGULARMESH METHODSFOR THE NEUTRONTRANSPORTEQUATION, Los Alamos Report LA-UR-73-479.

[32] B. Cockburn, C.-W. Shu, The runge-kutta discontinuous galerkin method for conservation laws v: Multidimensional systems, Journal of Computational Physics 141 (2) (1998) 199–224.

[33] K. Bey, J. T. Oden, A runge-kutta discontinuous finite element method for high speed flows, in: AIAA Computational Fluid Dynamics Conference, 10 th, Honolulu, HI, 1991, p. 541–555.

[34] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution of the 2D euler equations, Journal of Computational Physics 138 (2) (1997) 251–285.

[35] T. Toulorge, W. Desmet, CFL conditions for runge-kutta discontinuous galerkin methods on triangular grids, Journal of Computational Physics 230 (12) (2011) 4657–4678.

[36] H. L. Atkins, C.-W. Shu, Quadrature-free implementation of discontinuous galerkin method for hyperbolic equations, AIAA Journal 36 (5) (1998) 775–782.

[37] M. Dumbser, C.-D. Munz, Building blocks for arbitrary high order discontinuous galerkin schemes, Journal of Scientific Computing 27 (1-3) (2005) 215–230.

[38] M. Dumbser, D. S. Balsara, E. F. Toro, C.-D. Munz, A unified framework for the construction of one-step finite volume and discontinuous galerkin schemes on unstructured meshes, Journal of Computational Physics 227 (18) (2008) 8209–8253.

[39] E. M. Ronquist, A. T. Patera, Spectral element multigrid. i. formulation and numerical results, Journal of Scientific Computing 2 (4) (1987) 389–406.

[40] M. Zhang, C.-W. Shu, An analysis of and a comparison between the discontinuous galerkin and the spectral finite volume methods, Computers and Fluids 34 (4-5) (2005) 581–592.

[41] Y. Sun, Z. J. Wang, Evaluation of discontinuous galerkin and spectral volume methods for scalar and system conservation laws on unstructured grids, International Journal for Numerical Methods in Fluids 45 (8) (2004) 819 – 838.

[42] Y. Guo, A. T. Hsu, J. Wu, Z. Yang, A. Oyediran, Extension of CE/SE method to 2D viscous flows, Computers & Fluids 33 (10) (2004) 1349–1361.

[43] S.-C. Chang, Courant number and mach number insensitive CE/SE euler solvers, Tucson, Arizona, 2005, p. 45.

[44] C.-L. Chang, Three-dimensional navier-stokes calculations using the modified space-time CESE method, Cincinnati, Ohio, 2007.

[45] S.-C. Chang, S.-T. Yu, A. Himansu, X.-Y. Wang, C.-Y. Chow, C.-Y. Loh, The method of space-time conservation element and solution element – a new paradigm for numerical solution of conservation laws, in: Computational Fluid Dynamics Review 1998, m. hafez and k. oshima, eds. Edition, World Scientific, New Jersey, USA, 1999.

[46] B. Wang, H. He, S.-T. J. Yu, Direct calculation of wave implosion for detonation initiation, AIAA Journal 43 (10) (2005) 2157–2169.

[47] D. L. Bilyeu, Numerical simulation of chemical reactions inside a shock-tube by the space-time conservation element and solution element method, Masters thesis, Ohio State University (2008).

[48] D. L. Bilyeu, S. T. J. Yu, D. Davis, Simulation of shock-tube flows with detailed finite-rate chemistry by the CESE method, Vol. 2009-0811, AIAA, Orlando, FL, 2009.

[49] S. Chang, C. Loh, S. Yu, Computational aeroacoustics via a new global conservation scheme, in: Fifteenth International Conference on Numerical Methods in Fluid Dynamics, 1997, pp. 159–165.

[50] C. Loh, L. Hultgren, S. Chang, Wave computation in compressible flow using space-time conservation element and solution element method, AIAA Journal 39 (5) (2001) 794–801.

[51] M. Zhang, S.-T. J. Yu, S.-C. H. Lin, S.-C. Chang, I. Blankson, Solving the MHD equations by the space-time conservation element and solution element method, Journal of Computational Physics 214 (2) (2006) 599–617.

[52] M. Zhang, S.-T. J. Yu, S.-C. Lin, S.-C. Chang, I. Blankson, Solving magnetohydrodynamic equations without special treatment for divergence-free magnetic field, AIAA Journal 42 (12) (2004) 2605–2608.

[53] J.-R. Qin, S.-T. J. Yu, Z.-C. Zhang, M. C. Lai, Simulation of cavitating flows by the space-time CESE method, 2001.

[54] J. R. Qin, S.-T. J. Yu, M.-C. Lai, Direct calculations of cavitating flows in fuel delivery pipe by the space-time CESE method, Journal of Fuels and Lubricants, SAE Transaction 108 (2001) 1720–1725.

[55] L. Yang, R. L. Lowe, S.-T. J. Yu, S. E. Bechtel, Numerical solution by the CESE method of a first-order hyperbolic form of the equations of dynamic nonlinear elasticity, ASME Journal of Vibrations and Acoustics 132 (5) (2010) 051003.

[56] L. Yang, Y.-Y. Chen, S.-T. J. Yu, Velocity-stress equations for waves in solids of hexagonal symmetry solved by the space-time CESE method, ASME Journal of Vibration and Acoustics 133 (2) (2011) 021001.

[57] Y.-Y. Chen, L. Yang, S.-T. J. Yu, Simulations of waves in elastic solids of cubic symmetry by the conservation element and solution element method, Wave Motion 48 (1) (2011) 39–61.

[58] X. Wang, C. Chen, Y. Liu, The space-time CE/SE method for solving maxwell's equations in time-domain, in: Antennas and Propagation Society International Symposium, 2002. IEEE, Vol. 1, 2002, pp. 164–167 vol.1.

[59] X.-Y. Wang, S.-C. Chang, A 2D non-splitting unstructured triangular mesh euler solver based on the space-time conservation element and solution element method, Computational Fluid Dynamics JOURNAL 8 (2) (1999) 309–325.

[60] S.-C. Chang, Courant number insensitive CE/SE schemes, in: 38th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Indianapolis, Indiana, 2002.

[61] S.-C. Chang, X.-Y. Wang, Multi-dimensional courant number insensitive CE/SE euler solvers for applications involving highly nonuniform meshes, in: 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Huntsville, Alabama, 2003.

[62] J. C. Yen, Demonstration of a multi-dimensional time-accurate local time stepping CESE method, in: 17th AIAA/CEAS Aeroacoustics Conference (32nd AIAA Aeroacoustics Conference), Portland, Oregon, June 5-8, 2011, Portland, Oregon, 2011.

[63] C.-L. Chang, B. S. Venkatachari, G. Cheng, Time-accurate local time stepping and high-order space time CESE methods for multi-dimensional flows using unstructured meshes, American Institute of Aeronautics and Astronautics, 2013.

[64] S.-C. Chang, Y. Wu, V. Yang, X.-Y. Wang, Local time-stepping procedures for the space-time conservation element and solution element method, International Journal of Computational Fluid Dynamics 19 (5) (2005) 359–380.

[65] Grant Cook, Zeng-Chan Zhang, Kyoung-su Im, Applications of the CESE method in LS-DYNA, in: 21st AIAA Computational Fluid Dynamics Conference, Fluid Dynamics and Co-located Conferences, American Institute of Aeronautics and Astronautics, 2013.

[66] S.-C. Chang, The a(4) scheme: A high order neutrally stable CESE solver, in: 43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference &amp; Exhibit, American Institute of Aeronautics and Astronautics, 2007.

[67] Y.-Y. Chen, A multi-physics software framework on hybrid parallel computing for high-fidelity solutions of conservation laws, Ph.D. thesis, The Ohio State University (2011).

[68] S.-T. Yu, S.-C. Chang, Treatments of stiff source terms in conservation laws by the method of space-time conservation element and solution element, Reno, Nevada, USA, 1997.

[69] A. Suresh, H. T. Huynh, Accurate monotonicity-preserving schemes with runge-kutta time stepping, Journal of Computational Physics 136 (1) (1997) 83–99.

[70] P. Woodward, P. Colella, The numerical simulation of two-dimensional fluid flow with strong shocks, Journal of Computational Physics 54 (1984) 115–173.

[71] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock capturing schemes II, Journal of Computational Physics 83 (1989) 32–78.

[72] M. Zhang, S. T. J. Yu, CFL number insensitive CESE schemes for the two-dimensional euler equations, AIAA, Orlando, FL, 2003.

[73] D. S. Balsara, C. Altmann, C.-D. Munz, M. Dumbser, A sub-cell based indicator for troubled zones in RKDG schemes and a novel class of hybrid RKDG+HWENO schemes, Journal of Computational Physics 226 (1) (2007) 586–620.

[74] J. C. Hardin, J. R. Ristorcelli, C. K. Tam, ICASE/LaRC workshop on benchmark problems in computational aeroacoustics (CAA), NASA Conference Publicatoin 3300, Hampton, Virginia, 1994.

[75] C. K. Tam, J. C. Webb, Dispersion-relation-preserving finite difference schemes for computational acoustics, Journal of Computational Physics 107 (2) (1993) 262–281.

[76] T. Suzuki, T. Adachi, S. Kobayashi, An experimental analysis on shock reflection over the two-dimensional model of a dust layer, in: AIP Conference Proceedings, Vol. 208, 1990, p. 776.

[77] A. Ambrosio, A. Wortman, Stagnation-point shock-detachment distance for flow around spheres and cylinders in air, J. Aerospace Sci 29 (7) (1962) 875.

[78] F. S. Billig, Shock-wave shapes around spherical-and cylindrical-nosed bodies., Journal of Spacecraft and Rockets 4 (6) (1967) 822–823.

[79] R. W. Dyson, Technique for very high order nonlinear simulation and validation, NASA/TM 2001-210985.